DEZEMBER 1984 ÖS 50,–/Sfr 6,– DM 6,–



Eine Markt & Technik Publikation

12|84

DAS MAGAZIN FÜR COMPUTER-FANS

Worauf man beim Kauf achten muß

Großer Vergleichstest: Joysticks

Mit wenigen Handgriffen umgerüstet

Die schnelle 1541

Print 64 – universelles Centronics-Interface

Druckeranschluß ohne Probleme

Neuer Kurs: Musik mit dem C 64

Test: Monitore

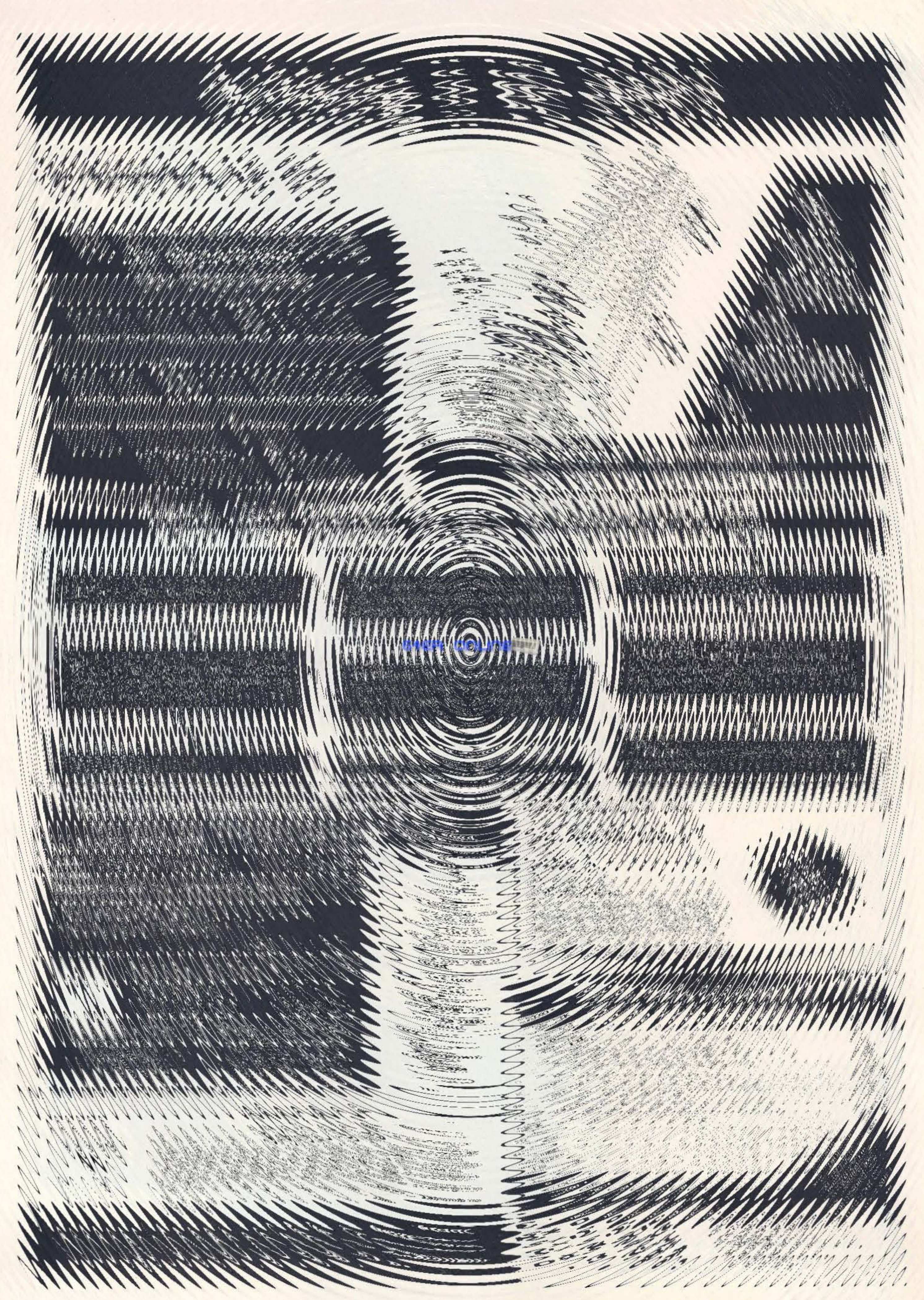
Listing des Monats

Super-Basic für den VC 20

So erweitern Sie das C 64-Betriebssystem

Was bringt Lernsoftware? ★ Soccer – das Spiel des Jahres ★ C 64 entwirft Kreuzworträtsel ★ Trace und Single Step für Assembler ★ VC 20-Datasette 10mal schneller ★ Tips & Tricks für den VC 20 und C 64





INHALT

Aktuell

Hardware-Test

Hardware

Software-Test

Software

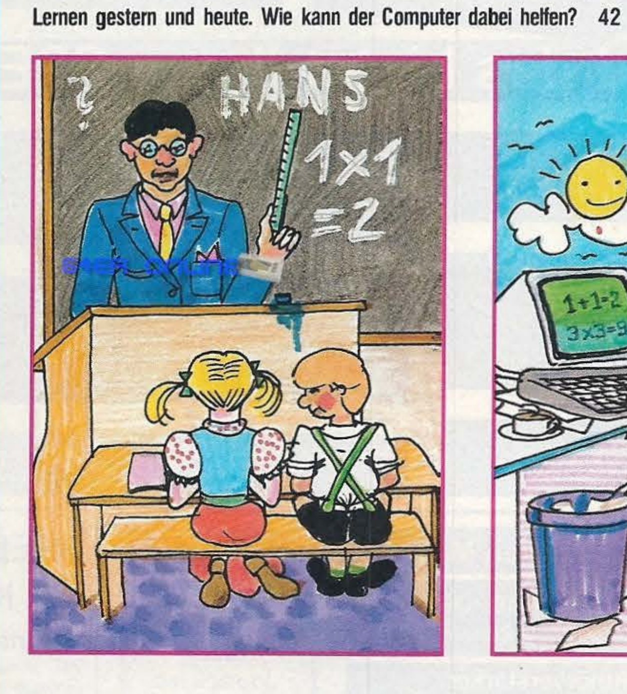
Spiele-Test

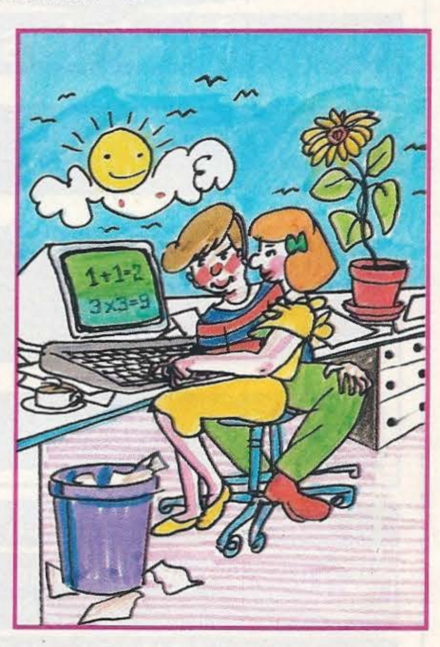
Wettbewerbe



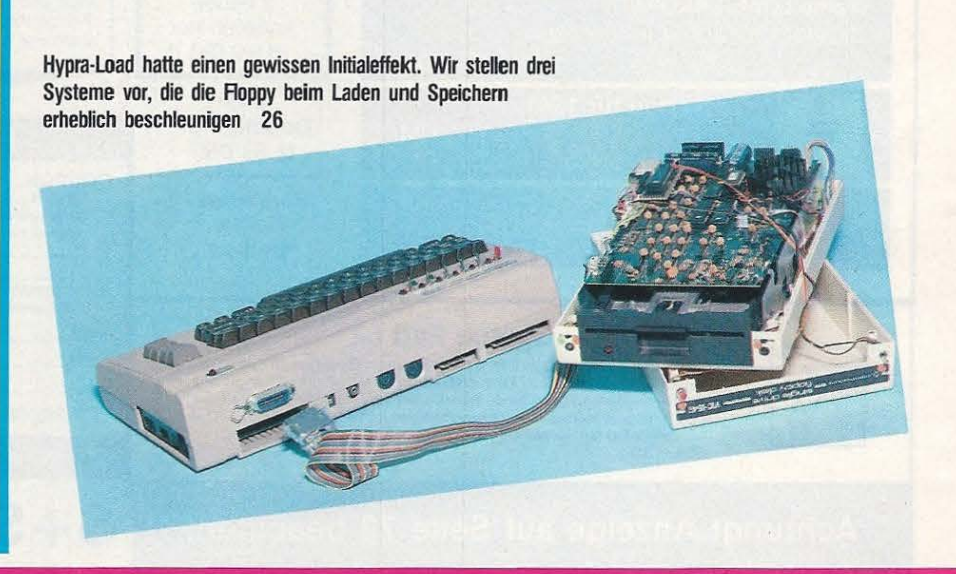
Graphic-Basic ist eine Spracherweiterung die sich nicht nur auf die Grafik bezieht 38

Lernen gestern und heute. Wie kann der Computer dabei helfen? 42





Hypra-Load hatte einen gewissen Initialeffekt. Wir stellen drei Systeme vor, die die Floppy beim Laden und Speichern erheblich beschleunigen 26









Programme zum Abtippen

Kurse

Rubriken



Mehr 64'er

Obwohl wir Ihnen in jeder Ausgabe des 64'er um die 120 Seiten redaktionellen Teil bieten, reicht der Platz nicht aus, um all das unterzubringen, was an interessantem und nützlichem Material vorhanden ist. Das hat natürlich auch seine positive Seite: Zeigt es doch, wieviel im »Volkscomputer« und in seinem größeren Bruder steckt — und daß der Benutzer nicht so schnell an die Grenzen des Systems stößt.

Da bei den Lesern eine starke Nachfrage nach Kursen besteht und die einzelnen Kurse ja auch in einer vernünftigen Zeit zu Ende gebracht werden sollen, haben wir versucht, zunächst hier Platz zu sparen: Der Kursteil hat eine »kompaktere« Form bekommen. Auf diese Art läßt sich pro Heft mehr Information unterbringen — ohne daß das (so glauben wir zumindest) auf Kosten der Übersichtlichkeit oder der Lesbarkeit geht.

Nach der Neugestaltung des Listingteils ist das ein zweiter Schritt, um die Wünsche unserer Leser noch besser zu erfüllen — ohne daß das ohnehin knapp kalkulierte Magazin dicker und damit teurer werden muß. Wahrscheinlich werden immer noch Wünsche offen bleiben — aber dafür gibt es ja die Karte »Lesermeinung«, auf der Sie (unter anderem) mitteilen können, was Ihnen an Informationen fehlt. Sie können sicher sein, daß wir die Karten fortlaufend sorgfältig auswerten — und schnell reagieren, wenn wir feststellen, daß bestimmte Themen besonders gefragt sind.

Michael Pauly, Chefredakteur

Aktuell



Interpod für C 64 und VC 20

Das in England entwickelte Interpod-Interface ist jetzt auch auf dem deutschen Markt erhältlich. Dieses Interface verfügt über einen IEEE-Bus zum Anschluß der Commodore-Peripherie mit dieser Schnittstelle, einen getrennten seriellen Ein- und Ausgang sowie eine RS232C-Schnittstelle.

Damit sind bis auf die Centronics-Schnittstelle alle wichtigen Verbindungen in einem Gehäuse vereinigt.

Das Interpod-Interface verfügt über einen eigenen 6502-Prozessor. Angeschlossen wird es an die serielle Schnittstelle vom C 64 oder VC 20. Angeboten wird das Interface bei Boston Computer zu einem empfohlenen Endverbraucherpreis von 398 Mark.

Info: Boston Computer, Rosenheimer Straße 145a, 8000 München 80, Tel. 089/ 491073/74

CP/M-CBM 64: Multi-Micro-Entwicklungssystem

Das Multi-Micro-Entwicklungssystem vereinigt die beiden Betriebssysteme Commodore-CBM und Digital Research CP/M 2.2 in einem Gerät. Es gestattet die Entwicklung verschiedener 8-Bit Mikroprozessor-Systeme mit Hilfe eines leicht modifizierten C 64. Die integrierte Programmiereinheit arbeitet mit interaktiver Bildschirmführung auf beiden Betriebssystemen und erlaubt Filetransfer, Duplizieren, Schnellprogrammierung, Anzeigen von Speicherinhalten, Vergleich zwischen Memory und EPROM, Erased-Test, mit frei definierbarem Memory- und EPROM-Bereich. Programmierbar sind zur Zeit die EPROM-Typen Intel 2758 bis 27256, Texas 2516 bis 2564, 2732, 2764. Außerdem die Single-Chip-Mikroprozessoren Intel 8741 bis 8749 beziehungsweise 8041 bis 8050 inklusive der H- und C-Typen.

Das System besteht aus CBM-CP/M-Zentraleinheit mit 64 KByte Speicher, 12 Zoll S/W-Monitor, 5¼-Zoll-Diskettenlaufwerk, EPROM-Programmiereinheit, EPROM-Löschgerät, Drucker, 40/80 Zeichen Bildschirmumschaltung und Tastatur.

Info: Ingenieurbüro für elektronische Systeme, Dipl.-Ing. K.E. Wnuk, Nansenstr. 3, 6100 Darmstadt, Tel. 06151/84439





Computer-Seminare

Cossem bietet in Nürnberg Kursprogramme für das Erlernen der Programmiersprachen Pascal, Basic, Fortran und Assembler 6502/6510 als auch einen »Anfängerkurs« (Wie funktioniert ein Computer?) und einen speziell auf den Commodore 64 ausgelegten Kurs an.

Info: Cossem, Computer-Software-Seminare, Gerlestraße 13, 8500 Nürnberg 40, Tel. 0911/262228.

Neue 128-KByte-ROM-Platine für den Commodore 64

Eine Platine für insgesamt 128 KByte ROM wurde von Frank Computertechnik, München vorgestellt. Die Platine ist für die Aufnahme von 8- bis 32-KByte-EPROMs vorgesehen. So können Speicherstufungen in 8-, 16-, 32-KByte-Schritten vorgenommen werden. Die Steuerung der einzelnen Speicherbereiche übernehmen zwei Register. Die Bedienung dieser Register wird in der Form eines Auswahlmenüs programmiert. Ein Directory mit dem Inhalt der Platine wird so angezeigt und das gewünschte Programm mit Knopfdruck gestartet. Da die Platine vollständig abzuschalten ist, sollte einer dieser Wahlpunkte auch aus dem Sprung in das normale Basic bestehen. Die Platine beeinflußt dann das Laden von Basic- und Maschinenprogrammen in keiner Weise. Ein Verlust an Speicherplatz tritt in diesem Fall nicht auf. Die Beschreibung zur Platine ist sehr ausführlich und geht auch auf verschiedene Programmierschritte (Autostartkennung, Betriebssystem- und Basic-Initialisierung) ein. Die Platine ist sehr solide aufgebaut und wird mit einem stabilen Gehäuse geliefert. Der Preis soll bei zirka 100 Mark liegen.

(Arnd Wängler)

Info: Frank Computertechnik, Metzstraße 8, 8000 München

Matrixdrucker von Olympia

Von Olympia gibt es jetzt aus der compact-Serie mit dem Olympia electronic compact NP auch einen Matrixdrucker. Dieser Drucker soll Epson FX-kompatibel sein und hat dazu noch weitere Eigenschaften. So kann per Tastendruck in Schönschrift gedruckt werden. Durch eine Gummiwalze wird ein sehr leiser Druck erzielt. Mit Bedientasten kann der linke und rechte Rand gesetzt werden. Der Traktor ist auf schmales Papier (einbahnige Etiketten) einstellbar. Der Preis beträgt für die Ausführung mit Centronics-Interface 1698 Mark, mit zusätzlicher V.24-Schnittstelle 1948 Mark. Die Ausführung für den C 64 mit Grafikfähigkeit kostet ebenfalls 1948 Mark.

Info: iti-Datentechnik, Telemannstraße 18, 7250 Leonberg, Tel. 0 71 52/63 05/7 10 74

Portables Meßdatenerfassungsystem mit SX 64

Der SX 64 kann, mit einem digitalem Speicheroszilloskop ausgesrüstet, zur Dastellung und Berechnung periodischer als auch transienter Signale in Industrie und Forschung eingesetzt werden.

Die Abtastrate beträgt 20 MHz bei 8 Bit, die Speichertiefe reicht von 4 bis 16 KByte und es sind sowohl Realtime- als auch Speicherbetrieb möglich. Es lassen sich beispielweise Impulssignale bis zu 6 MHz darstellen. Die Darstellung auf dem Oszilloskop ist flackerfrei und kann durch eine Lupenfunktion unterstützt werden.

Info: GTE, Brunnengräber Str. 6, 6362 Wöllstadt 2, Tel. 06034/3024/3025

Minimodem billiger

CTK hat die Preise für die Akustikkoppler der Serie Minimodem aus dem Hause Modular Technology um 25 Prozent gesenkt. Das CTK Minimodem 3005 (mit FTZ-Nummer) 300 Baud vollduplex kostet nun 585 Mark. Das neue Schwestermodell CTK Minimodem 3005 S (Originate/Answer umschaltbar) kostet jetzt 655 Mark.

Info: CTK Computer-Text- und Kommunikations-Systeme, Dolmanstraße 82, 5060 Bergisch Gladbach



Das tragbare Maßdatenerfassungssystem mit SX 64

Kurze Reparaturzeiten

Schnelle Reparaturen kündigt der offizielle Systemhändler für CBM-Geräte, die Firma HDS in München, an. Die Fachwerkstätte übernimmt Servicearbeiten an allen CBM- und VC-Geräten und will diese innerhalb kürzester Zeit fachmännisch ausführen. Wir haben die Probe gemacht und bekamen unser Gerät innerhalb eines Tages repariert zurück. Für Bastler bietet HDS zusätzlich den Verkauf von Commodore-spezifischen ICs (in kleinen Mengen).

Info: HDS Prüftechnik, Maria-Eich-Str. 1, 8000 München 60, Tel. (089) 83 7021-22



Kopfreinigungs-Diskette

Das Kopfreinigungs-Kit von Dysan besteht aus zwei Reinigungs-Disketten und einer Dosierflasche mit genügend Reinigungszyklen für 40 Reinigungsvorgänge (50% mehr als bisher). Mit dem Kit sollen Datenverlust, Systemausfall sowie Disketten- und Kopf-Verschleiß als Folge von Verschmutzungen vermieden werden. Als Besonderheit haben die Reinigungs-Disketten einen dünnen Film-Aufkleber über der Schreib-/Leseöffnung mit der bei einseitigen Laufwerken eine bessere Druckverteilung erzielt wird. Der Preis liegt bei 75 Mark ohne Mehrwertsteuer.

Info: Dysan, Frankfurter Allee 27-29, 6236 Eschborn, Tel. 061 96/48 1641



Interessant bis brisant – die elektroni

Würde die Tagesschau über Mailboxen berichten, Herr Köpke könnte jeden Abend neben einem Haufen neuer Telefonnummern Erstaunenswertes dem Publikum mitteilen. In der bundesdeutschen Mailbox-Szene tut sich allerhand; Grund genug, sich einmal genauer umzuschauen.

Was eine Mailbox ist, dürfte mittlerweile sicher hinreichend bekannt sein. Wie eine solche funktioniert und zu betreiben ist, hat sich wohl auch schon rumgesprochen, denn wie sonst würden diese elektronischen Briefkästen jeden Tag wie Kraut und Rüben aus dem Boden wachsen, der, nebenbei bemerkt, ausgerechnet im Industriegebiet rund um den Ruhrpott hierfür auch noch am fruchtbarsten erscheint.

Mailboxen gibt es seit der Erfindung der telefonischen Datenfernübertragung, doch erst preiswerte Geräte, wie zum Beispiel Akustikkoppler, lassen es zu, daß jetzt auch jeder seine eigene Box aufmachen kann. Voraussetzungen hierfür sind lediglich die FTZ-Nummer am Koppler und, falls galvanisch angeschlossen, also per Postmodem, auch noch eine spezielle für den Computer. Das benötigte Programm gibt es für Programmiergegner bereits im Handel.

Auf der anrufenden Seite bedarf es nur der Terminalsoftware, und schon steht einem die Welt offen. Was sich aber in dieser nun wirklich abspielt, reicht von banal über kurios bis hin zu Dingen, für die sich auch Verwaltungsdetektive der Post zu interessieren beginnen.

Doch vornweg: Die meisten Mailboxen dienen dem harmlosen Austausch von Informationen zwischen Computerfreaks aller Couleur: Da werden Adventure-Lösungen gesucht oder weitergegeben, Tips und Tricks für Hard- oder Software verraten, Termine für Clubtreffen vereinbart oder auch mal mitgeteilt, daß Herbert Bit und Renate Byte »ONLINE«, lies freundschaftlich und frisch verliebt zusammen sind.

Im Grunde genommen also eigentlich nichts anderes als ein Schwarzes Brett, elektronisch auf dem Bildschirm dargestellt und einer breiteren Hacker-Öffentlichkeit zugänglich gemacht. Zudem bieten einige Mailboxen, meist die, die von kommerziellen Anbietern oder Verlagen betrieben werden, noch die Möglichkeit, Programme in Form von Listings kostenlos abzurufen. Oft genug sind dies jedoch solche, die ohnehin schon hinreichend bekannt sind; man spart sich aber immerhin das mühselige Eintippen der Zeilen.

Doch es gibt auch andere, mehr verborgene Seiten in den Mailbox-Dateien. Bei einigen kann man sich direkt oder via Post als Benutzer eintragen lassen und so Nachrichten empfangen, die nur für sie oder ihn persönlich bestimmt sind. So ist ein relativer Schutz der Daten, die man weitergeben oder bekommen will, einigermaßen gewährleistet. Wer also in die höheren »Hackerweihen« im Mailbox-Verkehr einsteigen will, braucht demnach unbedingt einen Hakker-Namen und ein »geheimes« Password.

Wem hierzu nichts einfällt, kann sich ja fürs erste mal als GAST, GUEST, VISITOR oder sonstwas einloggen, das heißt melden; die meisten Mailboxen öffnen dann zumindest für den »öffentlichen« Teil der Nachrichten, Mailbox-Lesen und Mailbox-Eingeben, ihre Pforten. Und

Einige Mailboxen in Deutschland

```
x W

Sie sind der  10391. Anrufer !

 *****  *****  ****    ***    ****
   *    *      *   *  *   *  *
   *    ***    *   *  *****   ***
   *    *      *   *  *   *      *
   *    *****  ****   *   *  ****

Franzis-Teledatenservice  5.1

Ausg.stop=CTRL-S; dann  weiter=CTRL-Q, M
enue=CTRL-X
Inhalt von mc 8/84 ist schon abrufbar: A
ls Suchbegriff nach Befehl=f  '
<Return> einfach "84/8/" eingeben (Rubri
k MC 1984).
Bitte vergessen Sie bei Mailbox-Eingaben
 nicht, nach spaetestens etwa
70 Zeichen mit Return die Zeile abzuschl
iessen. Danke!
```

Schon sehr lange aktiv, Tedas von Franzis

```
                 *
*  D E C A t E S  INFORMATIONS - SYStEM
 Vers. III       :
*
                 :
:*:*: :*:****:*:*:* *:*****: :**:**:**:::
:***:*******:
SYS£EM-DATUM ==> 16.09.84
Ihr LOGON (NAME/PASSWORT) ==> ****:*

===  G A S T  ===
Wye Sie £INGETrAGENER-BENU£ZER werden
steht in Menue 1 / Punkt 4.
*:**: MItTEIL]NGEN VON DECAtES ****:

:**:: KEINE MITTEILNG FUER SIE :***:

BEDIGNUNGSANLEItuNG  ( J?N ) ==> n

SIND SIE SICHER ???

Mit welchem Befehl wird die ]ebertragung
 BEENDET ?
```

Ein humorvoller Sysop bei Decates

schen Briefkästen

spätestens beim Lesen dieser »MSGs«, Messages, kommen jedem bestimmt die Ideen für einen eigenen »LOGON« oder Codenamen. Hier einige Beispiele: Albatros, Hacman, Prof. Falken, Donald Duck, Blackbird, Sister CPU, Hacintosh, Dr. Binaer, Mr. Bit & Dr. Byte, Codo, Interram oder Hein-Soft. Namen gibt es noch und nöcher.

Einige findet man in geradzu jeder Box wieder, andere, wohl wegen der hohen Telefonkosten, nur in regionalen. Wiederum andere Teilnehmer verwenden einfach ihren »Vornamen aus Dingsbums«. Und manche haben anscheinend ihre Namen und Erfahrungen im Bereich des CB-Funks gesammelt.

Doch schauen wir uns die Mailboxen und deren Inhalt doch einmal etwas genauer an. Zwei Gruppen lassen sich auf Anhieb ausmachen: Die, die jede Nachricht erst nach Zensur »reinlassen«, das heißt veröffentlichen, und die, die (fast) ohne Zensur arbeiten. In beiden Gruppen geht's quer durch den Garten. Doch nur in letzteren findet der »PHREAK«, gleich Kunstwort aus »PHone« und »Computer fREAK«, die Informationen, die sein Herz höherschlagen lassen werden. Allerdings diese dann oft in einer, für den Telekommunikations-Anfänger und -Laien gänzlich unverständlichen Abkürzungssprache.

Wie gesagt, befinden sich die meisten Mailboxen im Telefonbereich »02xxx«, und dort beginnt auch unser Überblick. Die Telefonnummern der Mailbox, deren genauer Namen und ihre Betriebszeiten sind im Kasten ausführlich aufgeführt. Falls dieser nicht vollständig ist, freut sich die Redaktion über jeden eingesandten Nachtrag.

Und noch eine Bitte: Benutzt die Mailboxen nur zu den angegebenen Öffnungszeiten. Entweder wird sonst der normale, kommerzielle Betrieb tagsüber gestört oder, bei »Handvermittlung«, wo der SYSOP (SYStem OPerator) sich erst mit VIOCE meldet, um dann den Hörer auf den Koppler zu quetschen, ist es gut möglich, daß Ihr arme, nicht hackende Menschen aus ihrem wohlverdienten Schlaf klingelt.

Wenn eine große, öffentlich-rechtliche Institution wie der Westdeutsche Rundfunk, WDR, sich dem Thema Computer annimmt, ist man ja schon froh, daß in den »alten Medien« überhaupt etwas mit den »neuen« geschieht. Und Dank dieses »WDR-Computer-Clubs« gibt es auch eine gleichnamige Mailbox. Leider dachte man zu wenig an die Anrufer, die nur mit 40 Zeichen auf dem Bildschirm arbeiten können, und so ist die nicht umschaltbare 60-Zeichen-Anzeige manchmel etwas mühselig zu lesen. Inhaltsmäßig muß eine solche »offizielle« Mailbox wohl sehr aufpassen. Hauptsächlich beschränkt sich dieser meist auf liebe Grüße und harmlose Tips.

So ziemlich das gleiche Bild liefern auch die beiden Ortskonkurrenten »Epson« und »Software-Express«. Erstere erzählen viel von ihren eigenen Modellen und letztere, trotz ihres »guten« Rufes in der Hacker- und Cracker-

```
Der   W D R  -  COMPUTLRCLUB
begruesst Sie mit einem
[illegible] 9753  Transdata-Computer
>

**********************************************
**********
*   W D R - Computer(lub  Kommunikations
system    *
*   KOMCOM - 1   Tel.-Nr.: 02 21 - 37 10
 76       *
*   Version 1.0  von Guenter Eisbach  (C
) 1984    *
*   Timeout:  30 Sekunden
          *
*---------------------------------------------
---------*
*   Bedienung
sanleitung:                          *
*     RETURN  -  schliesst jede Eingabe a
b         *
*     Ctrl/S  -  haelt  die Ausgabe an
          *
```

Gut, aber schwierig zu lesen, die WDR-Mailbox

```
N.C.S.

(FAST) UNZENSIERTE MAILBOX RAUM KIEL
   Hallo und Willkommen bei N.C.S.
```

NCS ist eine der interessantesten Mailboxen

```
 **************************************
            O T I S  V2.1
             A G M C F +
  HELLMUT LANDOLT  COMMODORE-COMPUTER
 **************************************
Sie sind der 4581. Anrufer!
Benutzernr.? (0 = Erklaerung)? 0

Haben Sie schon eine Benutzernummer?
Wenn nicht, dann tippen Sie:
'neu'
Sie bekommen dann eine zugeteilt. Wenn
Sie nur mal 'reinschauen wollen,
schreiben Sie:
'gast'
Haben Sie Ihre Benutzernummer
```

Mailboxen als neues Medium für Werbung

Scene nun doch wieder wenig.

Wer's etwas intimer haben will, ruft doch mal um die Ecke an. Dort freuen sich die »C 64-Box«, die »Saturn-Box« und die der Firma »Symic« höchster Aktualität und Aktivität auf betreibender wie auf anrufender Seite. Dies kommt den Tips & Tricks genauso zugute wie den News, ohne die kein Hacker leben kann.

Immer noch im selben Vorwahlraum 02xxx tummeln sich seit neuestem die Boxen »Mythos« (heißer Atari-Tip), »Radio Schossau«, »Esprit«, »Computer Center CC EVD« und »Kobra«. Wer nicht gerade in der Nähe wohnt und Geld für teure Ferngespräche hat, wird dort die eine oder andere nützliche Info sicher finden. Ansonsten herrscht meist nur Lokalkolorit vor.

Das gleiche gilt übrigens auch für die Inselstadt Berlin. Trotz schöner Anlagen, auf denen die T.I.C. und die »MB-Berlin« betrieben werden, gibt es in anderen Mailboxen, pardon Berlin, einfach »heißere« Informationen und News.

Nächste Station: Hamburg. Doch das Tor zur Welt verhält sich, was seine Mailboxen angeht, wie das eigene Wappen — zu. Entweder ist wirklich dauernd besetzt oder stundenlang nur Quatsch auf dem Bildschirm. Denn die beiden Boxen, »Uni-Hamburg« und »M.C.S« gelten bekanntlich als die Haus-Boxen des Chaos-Computer-Clubs. Jenen Jungs also, die sich selbst zu den Oberhackern der Nation gekürt haben. Und dementsprechend sind auch die meisten Anfragen aus der ganzen BRD in diesen Boxen. Antwort gibt's leider nur selten.

Weiter nördlich wird's dann wieder lustig bis heiß. In Pinneberg eröffnete vor kurzem die »Wang-Info« (versuchsweise) im 24 Stundenbetrieb ihre elektronischen Pforten. Zwar ist die Teilnehmerzahl verständlicherweise noch gering, doch immerhin! Zudem läßt ein Helpmenü von 40 KByte auf vielfältige Entwicklungsmöglichkeiten schließen.

Ganz heiß wird's dann in Kiel. Dort ist die »N.C.S.-Box« zu Hause und tourjour zu erreichen. Wer sich da als User eintragen läßt, hat auch wirklich was davon.

Höchst informativ wird es dann erst wieder rund um die Main-Metropole. Bei »Decates« lohnt sich das Reinschauen allein schon wegen der witzigen Kommentare des »Sysop«. Ganz zu schweigen von der internationalen Mailbox-Nummernliste und den vielen informativen Einträgen der User. Als eine der wenigen bietet »Decates« auch ein »Download« von Programmlistings an.

Gleich nebenan ist »Tecos« mit einer ebenfalls sehr guten Box beheimatet. Die »Taunus-Box« mag wohl keiner, denn sie ist leider leer. Ganz anders dagegen die »Otis-Box«. Sehr interessant und sehr aktuell. Eine private Box, die auch Werbung und Angebote führt.

Im Süden ist dann wieder krasse Ebbe. Hier scheint's, kriegen die Jungs, trotz mannigfaltigen Ankündigungen, einfach kein Bein aufs Mailbox-Land. Einzige und nicht gerade aufregende Ausnahme ist die vom Franzis-Verlag betriebene Box »Tedas«. Ebenfalls mit Programm-Service und viel Eigenwerbung für das Verlagsprogramm. Nachrichten erscheinen im Bulletin erst nach vorheriger Zensur durch die Redaktion und sehen dann entsprechend brav aus.

Würde es Noten geben, nur einige wenige würden das Prädikat »sehr gut« bei allen Kriterien verdienen. Manche sind einfach beim besten Willen nicht benutzerfreundlich, andere zu lahm oder zu bieder. Doch wir stehen hier in diesem unseren Lande ja erst am Anfang einer Mailbox-Karriere.

(Klaus Koch/aa)

Telefon	Name	Zeit
0201-237396	Radio Schossau	22-10 Uhr
0211-414579	Software-Express	ab 18 Uhr
0211-593453	Epson	ab 18 Uhr
0211-328249	EDV	?
02151-801339	C 64-Box	ab 18 Uhr
02161-200928	Symic	?
0221-371076	WDR	00-24 Uhr
0221-1616284	Saturn	ab 18 Uhr
02202-50033	Computer Center	ab 18 Uhr
0231-779620	Mythos	ab 18 Uhr
02331-16401	Kobra	?
02841-66241	Esprit	?
030-7115078	T.I.C.	ab 19 Uhr
030-3052635	MB-Berlin	17-09 Uhr
040-41233098	Uni-Hamburg	20-06 Uhr
040-6523486	M.C.S	00-24 Uhr
04101-23789	Wang-Info	00-24 Uhr
04348-7513	N.C.S.	00-24 Uhr
06081-9677	Taunus	?
06154-51433	Decates	ab 18 Uhr
06181-48884	Otis	ab 18 Uhr
069-816787	Tecos	20-07 Uhr
0711-519008	Pluto	?
089-596422	Tedas	00-24 Uhr
089-598423	Tedas	00-24 Uhr



Die Zwangspausen des C 64

Bei längerem Suchen in einer Indexsequentiellen Datei macht der C 64 Pausen bis zu zirka fünf Minuten (bei Datamat bis zu 20 Minuten!). Woher kommen die Pausen, und wie sind sie zu beseitigen? *Henry König*

Wenn der C 64 sehr viele Strings zu verarbeiten hat, muß er ab und zu seinen dafür benötigten Speicherplatz aufräumen. Das heißt, er überprüft, welche Variablen noch aktuell sind und benötigt werden. Den Rest schmeißt er weg. Die dadurch entstandenen Lücken werden durch die verbleibenden Werte geschlossen. Diese Vergleiche und Verschiebearbeiten sind der Grund für diese langen Wartezeiten. Den Vorgang selber nennt man auch Garbage Collection. Man kann die Wartezeit verringern, indem man ab und zu ein FRE(0) ausführt. Dann wird automatisch eine Garbage Collection ausgeführt. Leider geht das nicht mit Programmen, die nicht unterbrochen werden können und die ein Garbage Collection nicht automatisch verhindern.

Ungewünschte Programme von Diskette?

Wenn ich ein Programm von Floppy lade, kommt manchmal ein Programm, das gar nicht gewünscht wurde. *Thomas Stepputat*

Abgesehen davon, daß Ihre Floppy vielleicht defekt ist, kann es andere Gründe für dieses Verhalten geben. Setzen Sie zum Beispiel Wildcards ein, das sind die Zeichen »*« und »?« zum Abkürzen von Namen oder Ersetzen von Zeichen, so kann es natürlich vorkommen, daß ein File geladen wird, das mit den gleichen Zeichen anfängt wie das gewünschte, jedoch vor diesem im Directory steht. Eine andere, jedoch ziemlich unwahrscheinliche Ursache könnte in der Verbindung VC 1541 und den von Ihnen angegebenen Drucker VC 1526 liegen, der ja bekanntlich den seriellen Bus stören kann. Allerdings ist uns diese von Ihnen beschriebene Wirkung noch nicht vorgekommen.

Fragen Sie doch!

Selbst bei sorgfältiger Lektüre von Handbüchern und Programmbeschreibungen bleiben beim Anwender immer wieder Fragen offen. Viel mehr Fragen ergeben sich bei Computer-Interessenten, die noch keine festen Kontakte zu Händlern, Herstellern oder Computerclubs haben. Sie können der Redaktion Ihre Fragen schreiben oder Probleme schildern (am einfachsten auf der beigehefteten Karte). Wir veranlassen, daß die Fragen von einem Fachmann beantwortet werden. Allgemein interessierende Fragen und Antworten werden veröffentlicht.

C-Compiler für C 64?

Ich suche dringend einen »C«-Compiler für den C 64. Wer kann helfen? *Hermann Eisele*

LCD-Display für C 64?

Läßt sich an den Commodore 64 ein LCD-Display anschließen? Wer hat eine Bauanleitung oder kann Hinweise geben? Wo bekomme ich ein für diese Zwecke brauchbares LCD-Display? *Thomas Müller*



MPS 802 baugleich mit VC 1526?

Ich habe eine C 1541 mit der ROM-Version 5 (neueste Version). Kann ich den Drucker 1526 ohne Probleme betreiben? Ist der MPS 802 absolut baugleich mit dem 1526? *Andreas Linnebach*

Betrachtet man die Hardware des MPS 802, stellt man eine weitgehende Baugleichheit zum 1526 fest. Das bezieht sich auch auf das identische Betriebshandbuch. Wenn Sie beim 1526 die ROM-Version 7C besitzen (die neueste Version), gibt es keine Probleme. Der MPS 802 besitzt das gleiche ROM. Ihre Version können Sie mit dem Drucker-Selbsttest feststellen.

Mailbox mit C 64?

Wer kann mir Telefonnummern nennen, unter der ein C 64 eine Mailbox bedient? *Stephan Prinz*

Spektral-Analyse mit C 64?

Ich möchte meinen C 64 als Spektral-Analyser einsetzen. Bisher konnte ich keine entsprechende Erweiterung finden. Wer hilft? *Emil Kubeck*

Deutsche Anleitung für MPS 802?

Ich brauche unbedingt eine deutsche Anleitung für den MPS 802.
Peter Sintke, Akazienweg 4, 8343 Triftern

Eigentlich sollte es Sache Ihres Commodore-Händlers sein, Ihnen eine deutsche Anleitung zu besorgen. Aber vielleicht hilft Ihnen ein(e) Leser(in).

Textverarbeitung mit DIN-Tastatur und deutschen Umlauten?

Gibt es für den C 64 Textverarbeitungsprogramme, die eine deutsche DIN-Tastatur auf dem Commodore ermöglichen mit Ausdruck der deutschen Umlaute? *Joachim Ludwig*

Es gibt sogar mehrere Programme, die das (fast) können. SM-Text hat einen kompletten deutschen Zeichensatz und auch y und z sitzen an der richtigen Stelle. Das gilt auch für den Textomat. Vizawrite setzt y und z so, wie sie auf der C 64 - Tastatur zu sehen sind. Blind schreiben ist also bei allen Programmen ohne Probleme möglich.

RENEW mit einem POKE?

Mit dem POKE 2050,10 bekommt man Programme, die mit SYS 64738, mit Reset oder mit NEW gelöscht wurden, wieder! Nach der Eingabe einer nicht belegten Programmzeile und (RETURN) ist das Programm wieder da. *Oliver Becker*

Schön wär's ja, aber leider nicht ganz richtig. Zwar kann man das Programm mit diesem POKE wieder sichtbar machen, doch lauffähig ist es nicht mehr. Sobald in dem Programm eine Variable definiert wird, hat man nur noch Schrott vor sich. Auch Speichern nach dem »RENEW« hilft nichts mehr. Der POKE 2050,10 setzt nämlich die Pointer nicht mehr neu.

Eingaben von der Tastatur verhindern?

Wie verhindere ich Eingaben von der Tastatur (insbesondere die STOP - Taste)? Kann man das mit einem POKE-Befehl verhindern? *Klemens Högenauer*

Eingaben von der Tastatur verhindert man mit POKE 649,0 (in Speicherstelle 649 ist die Größe des Tastatur-Puffers gespeichert). Die RUN/STOP und RESTORE-Taste sperrt man durch POKE 808,227. Die Sperre wird gelöst durch POKE 649,10:POKE 808,237.

VC 20 zum C 64 umrüsten?

Ich möchte meinen VC 20 zum C 64 umrüsten. Geht das? Kann man die 3584 Byte für die Grafik beim VC 20 vergrößern? *Stefan Junghänel*

Der VC 20 kann nicht in dem Sinne zum C 64 umgerüstet werden, daß anschließend die C 64-Software darauf läuft (es sei denn, man baut eine komplette C 64-Platine in das VC 20-Gehäuse ein.



Roßmöller Datentechnik, Finkenweg 1, 5309 Meckenheim bietet jedoch eine 64 KByte-Karte mit einer Speicherorganisation wie beim C 64 an. Das bedeutet, die 64 KBytes RAM liegen »parallel« zu ROM- und I/O-Bereich, so daß man wie beim C 64 das Betriebssystem ins RAM kopieren und verändern kann.

Die zweite Frage beruht offenbar auf einem Mißverständnis. Die 3583 Bytes sind nicht der Grafikspeicher, sondern der gesamte RAM-Bereich des VC 20 (lacht da jemand?), und der läßt sich natürlich durch Speichererweiterungen vergrößern.

Zeichensalat?

Es kommt sehr oft vor, daß ich nach dem Auflisten eines Programms nur noch merkwürdige Zeichen und Zeilen voller unsinniger Basic-Befehle sehe, wie zum Beispiel »10 PRINTPRINTSAVELISTGET ...« etc. Das Ganze wird zudem noch in vielen verschiedenen Farben angezeigt, teilweise auch in reverser Darstellung. Ich kann diese Zeilen dann auch nicht überschreiben. Woran liegt das, wie bekomme ich das weg ?

Dirk Schepanek

Der beschriebene Effekt kann drei verschiedene Ursachen haben:

1. Sie haben ein Maschinenprogramm ohne Basic-Vorspann geladen und dabei die Sekundäradresse 1 vergessen.
2. Sie haben ein Basic-Programm, das den unteren Basic-Bereich Speicherplatz für Grafik benötigt, einfach geladen, ohne mit bestimmten POKE-Befehlen den Basic-Start hochzusetzen.
3. Sie haben durch mehr oder weniger wildes POKEn in von Basic oder Betriebssystem benutzten Speicherstellen den Computer völlig irritiert.

Es gibt nun Gott sei Dank einen bemerkenswert einfachen Weg, diesen »Zeichensalat« wieder zu beseitigen: Durch Aus- und Wiedereinschalten des Computers.

Nochmals Rechengenauigkeit

In Ausgabe 10/84 antwortete Rolf Voigt auf die Frage, warum der C 64 bei »PRINT INT(3/0.03)« nicht 100, sondern 99 ausgibt. Leider war seine Antwort nicht ganz richtig.

Die selbstgestrickte Erklärung von Rolf Voigt für die Integer-Funktion (INT) ist zwar ganz pfiffig, aber falsch. Mathematisch gesehen ist die INT-Funktion eine Abbildung der Menge der reellen Zahlen auf die Menge der ganzen Zahlen, mit der Eigenschaft: INT(x) ist diejenige ganze Zahl, die kleiner oder gleich (!) x ist. Anders ausgedrückt schneidet INT einfach alle Nachkommastellen einer Zahl ab. Und genau das macht auch der C 64.

Warum erhält man dann aber die falsche Antwort »INT(3/0.03) = 99«? Dazu muß ich etwas ausholen. Testen Sie mal das folgende Programm:

```
10 K=0 : FOR I=0 TO 2 STEP 0.1
20 IF I=1 THEN PRINT "I=1 GEFUNDEN !" : K=1
30 NEXT
140 IF K=0 THEN PRINT "I=1 NICHT GEFUNDEN !"
50 END
```

Wenn Sie keinen Tippfehler gemacht haben und keine Sonderversion des C 64 besitzen, erhalten Sie als Ausgabe »I=1 NICHT GEFUNDEN!«. Erstaunlich! Wie kann so etwas sein? Wenn man von Null bis Zwei in Zehnteln zählt, erreicht man doch ganz sicher auch die Eins. Nun, der C 64 scheinbar nicht. Er stellt nämlich intern alle Zahlen binär dar, also auch solche kleiner als Eins. Fatalerweise ist es nun so, daß dezimal endliche Brüche in Binärschreibweise häufig nur als nicht abbrechende, periodische Zahlen geschrieben werden können. Das ist speziell bei 0.1 und auch bei 0.03 der Fall. Also wird die interne Binärdarstellung dieser Zahlen im Computer nach einer gewissen Stellenzahl abgebrochen (gerundet). Darum gibt der Ausdruck »3/0.03« beim Computer nicht ganz exakt 100, sondern etwas weniger.

Die Differenz können Sie sich mit »PRINT 100 - 3/0.03« selbst anschauen.

Wendet man nun die INT-Funktion auf eine Zahl an, die etwas kleiner ist als 100, dann kann nur 99 dabei herauskommen.

Das auf den ersten Blick merkwürdige Ergebnis »INT(3/0.01) = 99« ist also nicht, wie Herr Voigt meint, auf die INT-Funktion zurückzuführen (diese arbeitet völlig einwandfrei), sondern ist eine Folge der internen Binärdarstellung von Dezimalzahlen im C 64.

Aus diesem Grunde ist bei »IF...THEN« Abfragen auch Vorsicht geboten. So manches selbstgeschriebene Programm arbeit in manchen Situationen scheinbar nicht korrekt, weil einfach nicht an die nur begrenzte Rechengenauigkeit eines Computers gedacht wurde.

Jürgen Busch



Monitor-Anschluß

Frage: Wie schließe ich diverse Monitore an den C 64 oder VC 20 an?
Ausgabe: 8/84
(verschiedene Fragesteller)

Einen Schwarz-Grün Monitor schließt man an, indem man aus dem Video-Port des C 64 die Drähte »GND» und »VIDEO HIGH» mit den entsprechenden Monitoreingängen verbindet. Dabei muß »GND» am Monitor-Stecker außen liegen.

Einen RGB-Monitor anzuschließen ist nicht oder nur mit erheblichem Aufwand möglich. Mir ist kein Konverter bekannt; ein solcher würde im Selbstbau aber mindestens 400 Mark kosten.

Andreas Roeschies

Geldfrage

Welche Disketten haben das günstigste Preis-/Leistungsverhältnis? Kann man Sprites drehen? Wo gibt es die billigste Hard-/Software? Kann man mit dem Heimcomputer Geld verdienen?
Ausgabe: 10/84

Hendrik Richter

1. Der Preis eines Zehnerpacks Disketten mittlerer Preislage liegt zwischen 55 bis 68 Mark. Dazu wäre noch zu erwähnen, daß ein Diskettenpack für 48,50 Mark genauso gut sein kann wie einer zu 69,80 Mark. Achten Sie unbedingt auf eine Lochverstärkung, die eine Abnutzung der Innenseite fast völlig ausschließt. Auch auf solche Kleinigkeiten wie eine ausreichende Anzahl beschriftbarer Etiketten sollte man achten. Es ist sehr unangenehm, wenn man seine Disketten einmal neu beschriften will, dazu aber keine Möglichkeit, sprich Etiketten, mehr hat. Normalerweise gehören in einen Zehnerpack mindestens 15 bis 20 Etiketten.

Kommen wir nun zur Qualität. Es gibt mehrere, für den Laien unerklärliche Abkürzungen auf der Diskettenverpackung. Da findet man SD/SS, DD/SS oder auch DS/DD. Die Standarddisk ist wohl SS/DD, also »Single Sided, Double Density«, was soviel heißt wie »einseitig geprüft, doppelte Aufzeichnungsdichte«.

2. Die Sprites des C 64 kann man nicht drehen. Man kann allerdings mehrere Sprites entwerfen, die jeweils im Muster etwas gedreht sind, und diese dann in einer geeigneten Reihenfolge hintereinander auf den Bildschirm bringen.

3. Hier kann man Herrn Richter nur empfehlen, einmal einen Blick in den Anzeigenteil des 64'er-Magazins zu werfen.

4. Natürlich kann man mit dem Heimcomputer Geld verdienen. Zum Beispiel kann man sein Wissen in Worte kleiden und als Manuskript an Zeitschriften senden, die das dann gerne abdrucken.

Oder wie wär's mit einem kleinen Programm? So mancher Top-Programmierer hat bestimmt mit einem Heimcomputer angefangen, seine ersten Ideen zu verwirklichen.

Marco Kelting

Horizontales Scrolling

Wie kann man beim C 64 ein horizontales Soft-Scrolling realisieren?
Ausgabe: 10/84
Christoph Bergmann

Sowohl horizontales als auch vertikales Scrolling ist punktweise möglich. Hier eine Übersetzung aus dem »Commodore 64 reference guide«:

»Der VIC-II-Chip unterstützt weiches Scrolling sowohl in horizontaler als auch in vertikaler Richtung. Weiches Scrolling ist eine punktweise Bewegung des gesamten Bildschirms. Es kann aufwärts, abwärts, nach links oder nach rechts gehen. Es wird verwendet, um neue Daten weich auf den Bildschirm zu bringen, während auf der anderen Seite Zeichen weich das Bild verlassen.

Während der VIC-II-Chip den größten Teil der Arbeit macht, müssen Sie das eigentliche Scrolling mit Hilfe eines Maschinenprogramms erzeugen. Der VIC-II-Chip hat die Fähigkeit, das Videobild in acht verschiedene horizontale und acht verschiedene vertikale Richtungen zu bringen. Diese Positionierung wird von den VIC-II-Scrolling-Registern überwacht. Der VIC-II-Chip hat auch einen 38-Zeichen- und einen 24-Zeilen-Modus. Der kleinere Bildschirm wird dazu verwendet, um Ihnen Platz zur Verfügung zu stellen, die neuen Zeilen zum Hereinrollen außerhalb des sichtbaren Bildschirms zu speichern. Die Vorgehensweise im einzelnen:

1. Bildschirm verkleinern (der Rand wird breiter).
2. Das Scrolling-Register auf Maximum oder Minimum einstellen, je nach Richtung.
3. Die neuen Zeichen in den nicht sichtbaren Bildschirmbereich schreiben.
4. Das Scrolling-Register vergrößern oder verkleinern, bis zum anderen extremen Wert.
5. Verwenden Sie an diesem Punkt eine Maschinenroutine, um den gesamten Bildschirminhalt, um ein Zeichen in die Scrollrichtung zu verschieben.
6. Weiter mit Schritt 2.

Für nähere Einzelheiten sollten Sie sich Literatur über den VIC-II-Chip beschaffen. Wenn Sie kein versierter Programmierer mit Maschinensprache-Kenntnissen sind, sollten Sie allerdings die Finger vom Scrolling lassen.

Holger Jakobs

Schwierigkeiten mit Softwareschutz?

Wieso ist das Programm auf einer geschützten Diskette nach einem »Backup» nicht lauffähig ?
Caronni Germano

Weil der Software-Hersteller durch den Kopierschutz gerade den »Backup« verhindern will. Zu diesem Zweck sind meist gleich mehrere Sicherungen eingebaut. Um es mit den Software-Herstellern nicht zu verderben, wollen wir Ihnen aber lieber nicht verraten, wie das im einzelnen funktioniert.

Simons Basic und Turbotape

Kann man Turbotape und Simons Basic zusammen benutzen?
Ausgabe: 8/84 *Rolf Lehr*

Leider kann Turbotape nicht mit Simons Basic laufen. Diese Funktion wird durch zwei Dinge verhindert: Erstens liegt Turbotape im Speicherbereich $C000-$CFFF, welcher auch von Simons Basic verwendet wird, und zweitens verbiegen beide Utilities die CHRGET-Routine im RAM, um die von beiden Erweiterungen zur Verfügung gestellten neuen Befehle decodieren zu können.

Marc Haber

Piraten, Piraten

Bezugnehmend auf die Kolumne »Piraten, Piraten« in der Ausgabe 8/84 kann ich Ihnen bestätigen, daß die Firma »R & S Computerorganisation« eine Schwindelfirma ist. Unter dem Aktenzeichen — 52 Js 1025/84 — hat die Staatsanwaltschaft beim Landgericht Berlin dies noch einmal ausdrücklich bestätigt. Man sollte aber auf jeden Fall die Leser darauf hinweisen, daß der Handel mit Raubkopien strafrechtlich relevant ist und zu hohen zivilrechtlichen Schadenersatzansprüchen führen kann. Keinesfalls sollte man aber ohne weiteres Unterlassungserklärungen unterschreiben, bevor man hinsichtlich des rechtlichen Sachverhaltens Klarheit hat. Einige Firmen scheinen tatsächlich darauf spezialisiert zu sein, mit überhöhten Schadensersatzansprüchen die Leute so einzuschüchtern, daß diese nachher freiwillig einen zwar geringeren, aber eigentlich nicht notwendigen Betrag zahlen.

Dr. H. G. Baare-Schmidt,
Rechtsanwalt

Wo gibt's den »Super Expander«?

Ich habe mir das Buch »Grafik mit dem VC 20« aus dem Markt & Technik Verlag gekauft. Dazu benötigt man aber eine Befehlserweiterung (Super Expander). Woher kann ich die bekommen?
Michael Schmitt

Die Grafikerweiterung VC 1211 A (auch als »Super Expander« bekannt) sollte bei jedem Commodore-Händler erhältlich sein.

Probleme mit Datasette

Ich habe andauernd »LOAD ERROR« bei meiner Datasette.
Ausgabe: 9/84 *Peter Ritter*

Bei Problemen mit Datasette hilft es oft, den Tonkopf mit Spiritus zu putzen (Wattestäbchen verwenden). Außerdem sollte man keine Chromdioxid-Kassetten verwenden. Auch die Justierung des Tonkopfes mittels der dafür gedachten Schraube bringt manchmal Erfolg.

Klaus Kohler

Drucker zu langsam?

Die an meinen C 64 angeschlossene Typenrad-Schreibmaschine »Privileg 3000«, betrieben mit Textomat von Data Becker ist mir viel zu langsam. Wer kann helfen? Und wie?
Jürgen Heesch

Typenraddrucker sind leider nun einmal um einiges langsamer als Matrix- oder Tintenstrahl-Drucker. Dieser Mangel ist Konstruktionsbedingt und läßt sich softwaremäßig nicht beheben. Es ist ähnlich wie mit den Autos der 18-PS-Klasse — sie sind einfach von Natur aus langsam. Die einzige Abhilfe besteht in der Anschaffung eines schnelleren Modells.

Testprogramm für C 64

Gibt es ein Testprogramm für den C 64?
Ausgabe: 10/84
Peter Strempel

Die amerikanische Firma M-W Dist. Inc., 1342B Route 23 Butler NJ07405 vertreibt das Programm »Mr Tester«.

Überprüft werden: Joystick, Speicher, SID, Bildschirmposition und -Farbe, Tastatur, Kassette (lesen/schreiben), Disk (Spur und Sektor lesen/schreiben), Diskettenformat und Drucker. Der Preis beträgt 29,95 Dollar.

David Twigg-Flesner

Reset-Schalter zerstört Computer?

Ich wollte einen Reset-Schalter in meinen C 64 einbauen lassen. Mein Händler sagt aber nein dazu. Frage: Stimmt es, daß nach einem Reset durch Kurzschluß einige Bausteine im Computer zerstört werden können?
Lothar Birkenstock

Sie sind einem Ammenmärchen aufgesessen. Ein ordnungsgemäß eingebauter Reset-Schalter kann nichts zerstören. Der Netzschalter Ihres Computers ist beispielsweise ebenfalls mit der RESET-Leitung des Prozessors verbunden, und niemand würde auf die Idee kommen zu behaupten, das Einschalten eines Computers könnte diesen zerstören.

Geht TI$ falsch?

Die eingebaute Uhr (TI$) verliert die richtige Zeit, wenn Daten oder Programme auf Kassette gespeichert werden. Nach Beendigung eines Abspeichervorganges hat die Uhr einen Sprung nach vorne getan, der jedoch leider keinerlei Beziehung zur tatsächlich vergangenen Zeit hat. Das ist bei meinem Programm-Thermometer mit stündlichem Abspeichern der Temperatur auf Kassette sehr störend.
Rolf Tulewski

Sowohl die interne Uhr (TI$) als auch die Kassettenoperationen werden bei VC 20 und C 64 über Interrupt gesteuert. Da es beim Laden und Speichern von Programmen und Daten auf hohe zeitliche Genauigkeit ankommt, wird die interne Uhr für die Dauer dieser Operationen einfach abgeschaltet. Nach Beendigung der Kassettenoperation kann TI$ daher einen nicht definierten Wert besitzen. Leider gibt es keine Möglichkeit, dies zu umgehen, es sei denn, man verwendet einen externen Zeitgeber.

Raubkopien testen?

Testen Sie doch mal Spiele, die noch nicht bei 80% aller Besitzer seit Wochen als Raubkopie vorliegen.
Willi Brechtel

Sie werden lachen, aber es ist oft so, daß Raubkopierer viele Spiele Wochen vor dem offiziellen Vertriebsbeginn bekommen. Sie werden verstehen, daß wir nicht gerne Spieletests von Raubkopien veröffentlichen. Es kommt nämlich manchmal vor, daß diese Kopien besser sind als die Originale und das könnte peinlich werden — nicht nur für uns.



Der PET-Emulator der Demodiskette

Was hat es mit dem Programm »PET-Emulator« auf der Demo-Diskette auf sich?
Ausgabe 10/84
Wolfgang Joachim

Ohne Frage ist die Dokumentation von Commodore auch hier wieder sehr mies, das Programm »Emulator« läßt sich jedoch trotzdem sehr gut verwenden. Es gibt inzwischen zwar schon eine große Auswahl an Programmen für den C 64, aber einige Probleme (zum Beispiel mathematischer Natur) sind noch nicht umgesetzt worden. Da bietet sich doch eine Übernahme der alten CBM 3032-Programme an.

Konstruktionsbedingt unterscheiden sich der 3032 und der C 64 aber stark in der Speicherbelegung, dem Speicherplatz und der Zeropage. Zum Beispiel liegt das Video-RAM des 3032 im Bereich von 32768 bis 33792. Der C 64 dagegen hat sein Video-RAM ab Adresse 1024 bis 2023 und benötigt zusätzlich noch die Information über die Zeichenfarbe. Der 3032 verfügt über 31 KByte freies RAM, der C 64 dagegen über 38 KByte.

Um nun 3032-Programme direkt und ohne Umschreiben auf dem C 64 laufen zu lassen, braucht man nur den Emulator von der Demo-Diskette laden und starten. Danach wird das betreffende 3032-Programm von Kassette oder Diskette geladen und ist ohne weiters sofort lauffähig. Das gilt sowohl für Basic- als auch für reine Maschinenprogramme.

Bei einer geringen Anzahl von Maschinenprogrammen kann der Emulator aussteigen, beispielsweise wenn auf ROM-Routinen zugegriffen wird, die der Emulator nicht unterstützt. Da hilft nur eines: ausprobieren.

Der Emulator enthält übrigens einen DOS-Manager, der dem DOS 5.1 der Demo-Diskette entspricht.

Alles in allem ist der Emulator eine nützliche Sache. Er vergrößert die ohnehin schon bemerkenswerte Programmvielfalt für den C 64 noch um einiges.

Stefan Ullmann

Wer kennt »Quicktext«?

Im 64'er-Magazin, Ausgabe 6/84, wurde das Listing zum Textverarbeitungsprogramm »Quicktext« abgedruckt. Seit Erscheinen dieses Heftes sitze ich nun daran. Selbst nach der Berichtigung im »Druckfehlerteufelchen« und nach wiederholtem Checken des Listings läuft das Programm einfach nicht. Wer kann helfen?
Margie Bastow

Simons Basic und DOS 5.1

Wie kann man gleichzeitig mit Simons Basic und DOS 5.1 arbeiten?
Ausgabe: 64/10/84
Detlev Preisler

Beide Programme arbeiten wie folgt zusammen:
— Simons Basic laden und starten.
— DOS 5.1 mit "LOAD" "DOS 5.1",8,1 laden und mit SYS 52224 starten.

Es geht natürlich auch umgekehrt, kann aber dann in einigen Fällen zu Problemen führen.

Hier nun die Abhilfen:
— Sind beide Programme im Speicher, dann wird das DOS mit SYS 52224 eingeschaltet.
— Werden DOS-Befehle nur mit einem müden READY beantwortet (oder überhaupt nicht), dann gibt man " #8" ein. Die »8« steht dabei für die Geräteadresse der Floppy.

Es gibt aber auch eine Simons Basic-Version II die von anfang an problemlos mit dem DOS 5.1 läuft.

Wollen Sie antworten?

Wir veröffentlichen auf dieser Seite auch Fragen, die sich nicht ohne weiteres anhand eines gutes Archivs oder aufgrund der Sachkunde eines Herstellers beziehungsweise Programmierers beantworten lassen. Das ist vor allem der Fall, wenn es um bestimmte Erfahrungen geht oder um die Suche nach speziellen Programmen beziehungsweise Produkten. Wenn Sie eine Antwort auf eine hier veröffentlichte Frage wissen — oder eine andere, bessere Antwort als die hier gelesene — dann schreiben Sie uns doch. Die Antworten werden wir in einer der nächsten Ausgaben publizieren. Bei Bedarf stellen wir auch den Kontakt zwischen Lesern her.

C 64-Platine und Tastatur kaufen

Wo kann man eine C 64-Platine kaufen?
Ausgabe: 10/84 *Eric Kratzin*

Wo kann man eine C 64-Tastatur kaufen?
Ausgabe: 10/84
Oliver Varoß

Der TKD (Technische Kundendienst) von Quelle müßte in der Lage sein, die gewünschten Teile zu einem vertretbaren Preis zu beschaffen. In Ihrem Fall schlage ich aber vor, Sie tun sich zusammen und teilen sich ein komplettes Neugerät.

Holger Jacobs

Wo gibt's Comal?

Ich habe mit großem Interesse den ersten Teil Ihres ausgezeichneten Comal-Kurses gelesen. Leider haben Sie nirgendwo geschrieben, wo man Comal 0.14 erhalten kann und was es kostet.
Jürgen Kruse

Hier sind zwei Bezugsadressen:
Comal User Group
Otkerstr. 34
8000 München 90

Comal Gruppe A. Knapp
Giersdorfer Str. 10
2800 Bremen 44

Bei beiden Adressen erhalten Sie Comal 0.14 auf Diskette einschließlich Kurzanleitung/Befehlsübersicht für 25 Mark Vorkasse (Schein oder Scheck). Für 30 Mark Vorkasse erhalten Sie zusätzlich noch das Handbuch. Wenn Sie zunächst nur das Handbuch haben wollen, kostet es 6 Mark. Bitte sehen Sie von Anfragen bei den beiden Comal-Gruppen ab, da alle anfallenden Arbeiten von den Mitgliedern in ihrer Freizeit ausgeführt werden müssen.

Programmunterbrechung bei Druckerausgabe

Bei der Druckerausgabe erhalte ich oft einen »DEVICE NOT PRESENT ERROR«, obwohl der Drucker eingeschaltet ist.
Ausgabe: 8/84
Rudolf Ott

Dieser Fehler tritt sehr häufig bei Druckern auf, die über den seriellen Bus des C 64 angeschlossen werden. Die Ursache dafür liegt in der Konzeption des seriellen Busses. Abhilfe: Statt PRINT# mit CMD arbeiten, dann tritt der Fehler nur noch äußerst selten oder gar nicht mehr auf.

Marc Haber

Wo gibt's EPROM-Brenner?

Wo kann ich den im Hardware-Test beschriebenen EPROM-Brenner von Roßmöller beziehen? *Jürg Kauer*

Die Bezugsadresse ist Roßmöller Datentechnik, Finkenweg 1, 5309 Meckenheim. Es wäre noch darauf hinzuweisen, daß Roßmöller die Preise erhöht hat und außerdem der »Bausatz« nur aus einer Platine besteht.



Die Scharfmacher

Um den Streit um den häuslichen Farbfernseher zu beenden, gibt es eigentlich nur zwei Lösungen. Entweder wird ein zweiter Fernseher gekauft oder gleich ein Datenmonitor. Wir stellen vier dieser Sichtgeräte vor.

Über technische Daten von Monitoren wird viel diskutiert, letztlich entscheidend über den Gebrauchswert eines Monitors ist aber immer die Qalität des Bildes. Diese ist aber von mehr als nur einigen Maßzahlen abhängig. Die meisten der in bunten Farbprospekten abgedruckten Daten verunsichern eher den Käufer, als ihm bei einer Entscheidung zu helfen. Oft wird von 40 oder 80 darstellbaren Zeichen pro Zeile gesprochen, oder mit Meßwerten nur so um sich geworfen. Dem Interessenten nutzt das sehr wenig.

Als wesentlichstes Beurteilungskriterium hat sich inzwischen die sogenannte Bandbreite (gemessen in Megahertz) eingebürgert. Die für eine deutliche Darstellung notwendige Bandbreite ist im wesentlichen davon abhängig, wieviele Einzelpunkte in einer Bildschirmzeile abgebildet werden. Bei[illegible] Commodore 64 sind das 730 Punkte in der Horizontalen. Die minimale Bandbreite für diese Punktdichte beträgt etwa 5 bis 6 Megahertz bezogen auf die erste Oberwelle. Das schafft sogar ein einfacher SW-Fernseher. Deutlicher wird die Darstellung allerdings erst, wenn eine höhere Bandbreite zur Verfügung steht. Einfache Monitore haben etwa zwischen 12 und 15, mittlere zwischen 15 und 20 und gute Monitore über 20 Megahertz Bandbreite. Bei einigen Studiomonitoren wird sogar eine Bandbreite von über 40 Megahertz benötigt. Wesentlich unsinniger ist die Angabe »Zeichen pro Zeile«.

Wer sich einen Monitor anschaffen möchte, sollte sich vorher überlegen, wofür er seinen Computer hauptsächlich verwenden will. Wer nur gelegentlich etwas programmieren, beziehungsweise spielen möchte, ist mit einem kleinen Farbfernseher sicherlich nicht schlecht bedient, zumal ein Fernseher ja auch als Zweitgerät nützlich ist. Wer aber höhere Ansprüche an die Bildqualität stellt, steht schnell vor der Frage, monochromer oder Farbmonitor? Für den Anwender, der selten mal ein Spiel- oder Grafikprogramm

Bild 1. Der Cable MC 3700

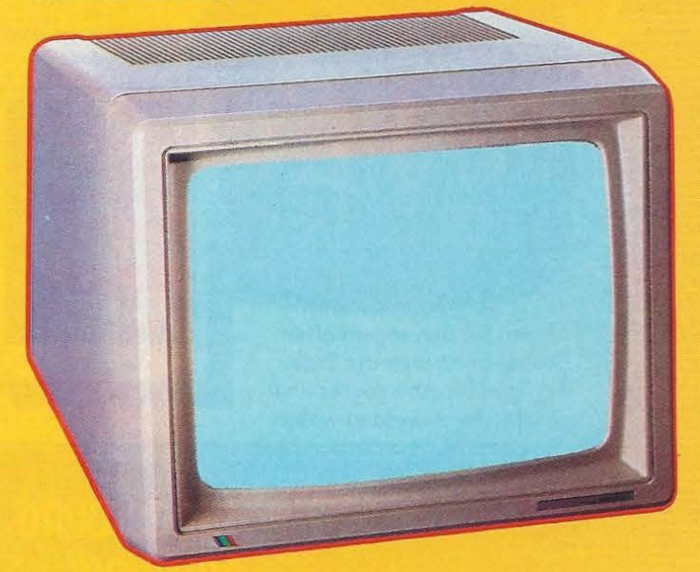


Bild 2. Der Cable besticht mit exzellenten Farben

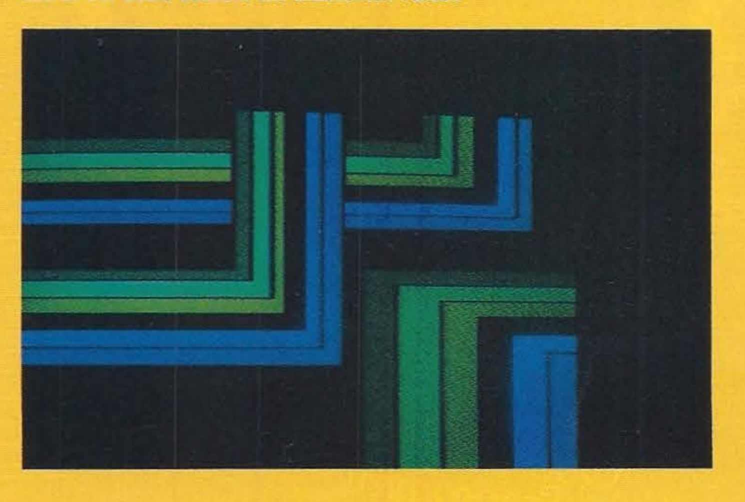




Bild 3. Nicht ohne Probleme: Der Taxan KX 12

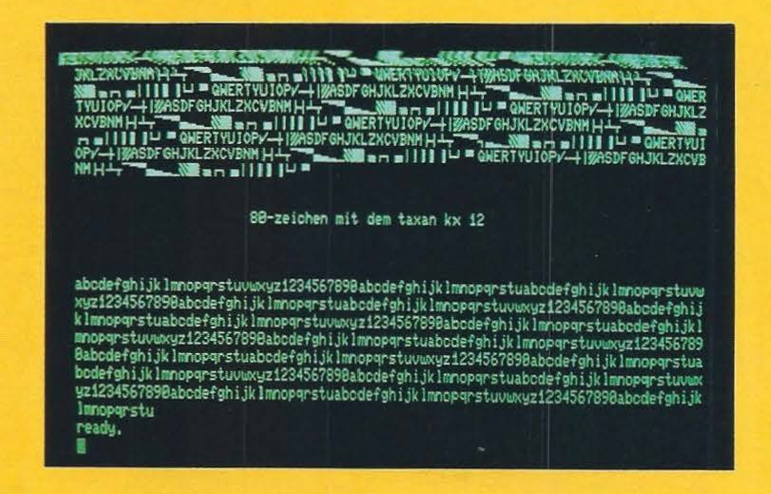


Bild 4. Die ausgezeichnete Darstellung von 80 Zeichen mit dem Taxan. Die Verzerrung in den oberen Zeilen konnten nicht beseitigt werden

verwenden möchte, haben wir drei monochrome Monitore verschiedener Qualitäts- und Preislagen getestet. Für die größte Überraschung sorgte aber der ebenfalls getestete Farbmonitor Cable MC 3700. Bei der Beurteilung der verschiedenen Monitore verwendeten wir zusätzlich die 80-Zeichenkarte von Decam.

Mit Tricks zum besseren Bild

Im Test zeigte sich, daß es gar nicht so einfach ist, dem Commodore 64 ein klares und kontrastreiches Bild zu entlocken. Zwar bieten der C 64 und der VC 20 auf ihrer Geräterückseite einen Videoanschluß an. Das dort bereitgestellte Videomischsignal (FBAS = Farb-, Bild-, Austast- und Synchronsignal) kann aber hohen Qualitätsansprüchen nicht genügen. Die besten Resultate lassen sich mit dem Weg erzielen, den Commodore bei seinen 1701/1702-Monitoren gegangen ist. Die zwei für das Bild zuständigen Signale Sync/Luminance (Synchronisation/Helligkeit) und Chrominance (Farbart und Farbton) werden extern im Monitor gemischt. Für den Anschluß eines monochromen Monitors ist es übrigens besser, nur das Sync/Lum-Signal zu verwenden, die Darstellung wird dann um einiges deutlicher.

Handbuch mit Fehlern

Die Abbildung der Video-Buchse im Commodore 64 Handbuch Seite 142 bezieht sich auf den VC 20. Beim Commodore 64 findet eine achtpolige DIN-Buchse Verwendung. Die Pinbelegungen stimmen aber im wesentlichen überein, lediglich der mittlere Pin (im Handbuch nicht abgebildet) trägt das Chrominance-Signal, die ebenfalls unterschlagenen Pins sieben und acht sind nicht angeschlossen. Für Besitzer eines Farbmonitors, wie dem Taxan Vision EX bleibt zum Anschluß eines Monitors nur das Video-Mischsignal. Trickreicher sind die Konstrukteure des Cable MC 3700 vorgegangen. Sie entnehmen dem Commodore die Signale getrennt und mischen sie erst im Monitor. Das Ergebnis ist beeindruckend. Das Bild zeichnet alle Konturen extrem scharf und verträgt selbst stärkste Farbunterschiede. Dabei sind die Farben sehr kräftig und leuchtend. Der Cable verfügt nicht nur über ein exzellentes Bild, sondern auch über einen guten Ton, der ebenso wie alle anderen Einstellungen an der Gehäusevorderseite geregelt wird. Ebenso wie seine Qualität, fällt auch das äußere Erscheinungsbild des Cable sehr positiv auf. Der Monitor wird komplett mit einem beweglichen Standfuß geliefert, der wie das Gehäuse beigefarbig ist. Der Cable ist ein gutes Beispiel für gelungenes Design und funktionelle Leistung. Lediglich der auf der Gehäuserückseite befindliche Netzschalter ist etwas schwer zugänglich. Die größte Überraschung erlebt man aber beim Aufschrauben des Cable. Normalerweise sind Produkte einer relativ unbekannten Firma mit Innereien aus dem Lande Nippons ausgestattet. Nicht so der Cable, er trägt auf allen seinen Bauteilen das Philips-Siegel. Bleibt die Frage, warum nicht gleich unter diesem Namen? Fürchtet Philips etwa die eigene Konkurrenz? Trotzdem ist der Cable ein sehr gutes Gerät, das als Farbmonitor sogar für eine erträgliche Darstellung von 80 Zeichen pro Zeile geeignet ist. Hierzu muß allerdings ein neues Kabel gelötet werden, denn die Decamp-Karte bietet ein BAS-Signal (monochrom) auf einem Chinchstecker an. Wer mit dem Cable ohne Farbe arbeiten will, findet auf der Rückseite des Gehäuses übrigens einen kleinen Schalter, der das Chrominance-Signal abschaltet. Schneller geht das allerdings mit dem Farbsättigungsregler.

Es muß nicht immer Farbe sein

Die drei zum Test zur Verfügung stehenden monochromen Monitore waren alle mit einer grünen Anzeige ausgestattet. Damit erschöpften sich aber schon weitgehend die Gemeinsamkeiten, denn es gab erhebliche Qualitätsunterschiede. Das erste Gerät war der Taxan KX 12 (Bild 3), der über eine Bandbreite von 20 Megahertz verfügt. Er wird mit grüner und bernsteinfarbener Farbträgerschicht sowohl für den deutschen, als auch für den amerikanischen Markt gebaut. Hierin liegt bereits sein größter Nachteil. Der von jedem Monitor benötigte Rasterfrequenzwert steht technisch bedingt in engem Zusammenhang mit der Frequenz des jeweiligen Stromnetzes (Deutschland 50 Hz, USA 60 Hz). Viele Umbau- und Anschlußschwierigkeiten rühren von diesem Umstand her. In der Regel ist es mit ei-

nem Austausch des Trafos im Netzteil nicht getan. Viele frequenzbestimmende Teile sind auf die 60 Hertz abgestimmt und funktionieren mit der Frequenz in Europa nicht mehr richtig. Das Ergebnis sind häufig Zittereffekte. Beim Taxan KX 12 treten diese Effekte kaum auf, dafür hat dieser Monitor bei einem sonst guten Bild eindeutig Schwierigkeiten, verzerrungsfrei zu bleiben. kaum, wohl aber in ihren Leistungsmerkmalen. Der Hersteller beider Monitore ist BMC. Der BM 12 ES für etwa 400 Mark (Bild 5) ist sowohl in grün, als auch in bernstein erhältlich. Seine Bandbreite wird mit 18 Megahertz angegeben. Die eigentliche Besonderheit dieses Monitors ist aber seine Filterscheibe. Direkt auf der Bildröhre wurde eine synthetische Folie aufgebracht, die für ist, gefiel der BM 12 EN ein wenig besser. Mit einem Preis von 425 Mark ist er allerdings auch einer der teuersten getesteten Geräte. Für den BM 12 EN wird eine Bandbreite von über 20 Megahertz angegeben. Das Bild ist sehr deutlich und absolut ruhig. Auch beim Einsatz der 80-Zeichenkarte wartete der BM 12 EN mit dem besten Ergebnis auf (Bild 8). Alle Zeichen waren gesto-



Bild 5. Auch optisch ansprechend — der BMC BM 12 ES

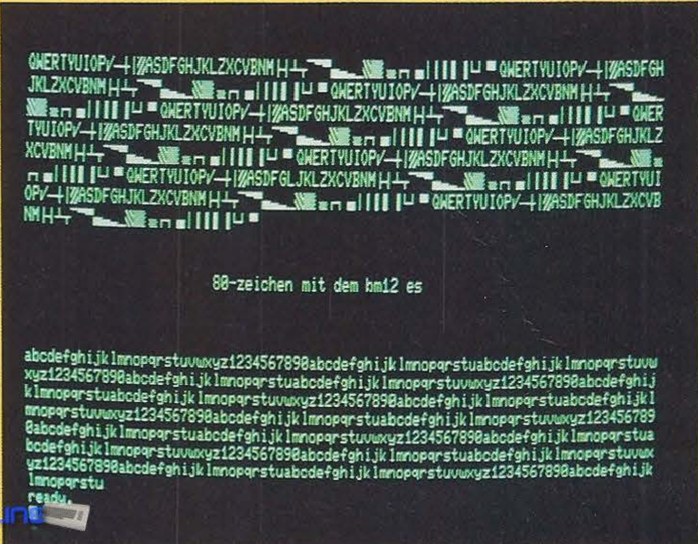


Bild 6. Ein gutes Beispiel für eine 80-Zeichen-Darstellung. Die leichte Krümmung der Zeilen ist auf die Bildschirmwölbung zurückzuführen.

Das erste getestete Gerät wies so starke Verzerrungen an den Rändern auf, daß eigentlich nur ein Transportschaden vorliegen konnte. Als aber, in etwas verminderter Form, das gleiche Problem auch beim zweiten Gerät auftrat, mußte auf einen Konstruktionsfehler geschlossen werden. Ganz besonders kraß fiel dieser Effekt bei Verwendung der Decam-Karte auf. Die Ränder verzerrten so stark, daß ein ordentliches Arbeiten kaum noch möglich war (Bild 4). Selbst die auf der Geräterückseite befindlichen, umfangreichen Einstellpotentiometer konnten nicht helfen. Obwohl mit einer hervorragenden Bandbreite von 20 Megahertz ausgestattet, hielt der zirka 400 Mark teure Taxan dem Test mit seinen Konkurrenten nicht stand. Auch wenn die eigentliche Zeichendarstellung gut war.

Die beiden anderen monochromen Testkandidaten sind eigentlich sehr nahe Verwandte, denn rein äußerlich unterscheiden sie sich ein blend- und reflexarmes Bild sorgen soll. Tatsächlich werden alle einfallenden Lichtquellen etwas diffuser, was bei längerem Arbeiten in stark erhellten Räumen als angenehm empfunden wird. Die Bildqualität des BMC BM 12 ES ist durchaus gelungen und die Ränder werden verzerrungsfrei abgebildet. Beim Einsatz der Decam-Karte waren die dargestellten Zeichen sehr gut lesbar und zeigten an den Zeichenrändern keinerlei Fadeout-Effekte (Bild 6). Auch im Dauereinsatz von mehreren Tagen wurde keine Verschlechterung der Bildqualität festgestellt.

Hervorragende Bildqualitäten

Obwohl zwischen zwei so guten Monitoren wie dem BM 12 ES und dem BM 12 EN (Bild 7) bei der Beurteilung der Bildqualität immer ein gewisser Anteil Subjektivität dabei chen scharf, so daß der Eindruck entstehen konnte, daß der Commodore 64 nie für weniger als 80 Zeichen pro Zeile konzipiert wurde. Auch beim BM 12 EN ist die Bildröhre entspiegelt, allerdings nicht durch eine Folie, sondern durch leichtes Anätzen der Oberfläche. Das Resultat fiel allerdings nicht so überzeugend wie beim ES-Modell aus.

Kein eindeutiger Sieger

Ist die Entscheidung zwischen einem farbigen- und einem monochromen Monitor erst einmal gefallen, so sind bis auf den Taxan Monitor alle getesteten Geräte durchaus empfehlenswert. Mit einem Preis von etwa 800 Mark zuzüglich 18 Mark für das Verbindungskabel ist der Cable MC 3700, der auch in einer RGB-Version geliefert wird, zwar nicht gerade billig, bietet aber von

allen bisher getesteten Farbmonitoren das beste Bild.

Ein monochromer Monitor sollte in der Regel dann eingesetzt werden, wenn auch eine 80-Zeichenkarte vorhanden ist, dabei ist es wichtig, daß die Zusammenarbeit mit einer solchen Karte problemlos funktioniert. Die beiden BMC-Monitore bieten hierfür die besten Voraussetzungen. Wer sich allerdings für alle späteren Anwendungen gerüstet wissen will, sollte die 25 Mark Mehrkosten aufbringen und sich gleich das bessere EN-Modell zulegen.

Die 80-Zeichen-Karte

Gute Dienste während des gesamten Tests leistete die 80-Zeichenkarte der Firma Decam. Wie schon im Monitortest angesprochen, sorgt sie für ein exzellentes Schriftbild in einer 5x8-Zeichenmatrix. Die Decam-Karte liefert zu einem Preis von 285 Mark aber nicht nur die notwendige Zeichenumdefinierung, sondern hat auch noch einige andere nützliche Funktionen. So wurden beispielsweise einige der Sonderbefehle des CBM 8032 implementiert. Alle Sonderbefehle werden über CTRL in Verbindung mit einer anderen Taste ein- beziehungsweise ausgeschaltet. Mit CTRL L und CTRL N wird beispielsweise zwischen dem Grafikmodus (sonst CBM + SHIFT) und dem Textmodus umgeschaltet.

Wesentlich interessanter ist aber die Fensterdefinition. Die Größe des Fensters wird durch die Position des Cursors bestimmt. Ist ein solches Fenster definiert, wirken sämtliche Befehle nur noch auf dieses Fenster. Erst durch zweimaliges Drücken der HOME-Taste wird die ursprüngliche Bildschirmgröße wieder hergestellt. Weitere Funktionen sind das zeilenweise Herauf- und Herabblättern des Bildschirms sowie das zeichen- und zeilenweise Löschen und Einfügen.

Wesentliches Kriterium für die Bewertung einer 80-Zeichenkarte ist ihr Grad der Kompatibilität. Bei der Decam-Karte ist der gesamte Basic-Speicher für den Anwender freigeblieben. Das bedeutet, daß alle Programme, die keine wesentlichen Veränderungen außerhalb des Basic-Speichers vornehmen, weiterhin verwendet werden können. Richtig interessant wird es bei der Frage, wie verschiedene Basic-Erweiterungen mit der Karte zusammenarbeiten. Die Bedienungsanleitung zur Decam-Karte gibt hier einige Hilfen. Möchte man beispielsweise Exbasic Level II mit der Karte verwenden, wird empfohlen: 1. Das Programm zu laden. 2. zu starten und 3. den Befehl SYS 49263 zu geben. Zusammen mit einem Expansionsstecker und dem Modul funktioniert die beschriebene Methode aber nicht.

Wesentlich besser gestaltet sich das Zusammenspiel mit der CP/M-Karte. Nach dem Laden eines kleinen Zusatzprogrammes kann tatsächlich mit der 80-Zeichenkarte und dem Modul zusammengearbeitet werden. Allerdings gelang es während des Tests nicht, eine Version des Textverarbeitungsprogramms »Wordstar« einzusetzen, da sich die Bildschirmausgabe bis auf die erste Zeile sperrte. Dieser Fehler muß aber nicht unbedingt auf die Karte zurückgeführt werden, denn wie jeder CP/M-Besitzer weiß, hat die CP/M-Karte ihren eigenen Willen. Problemlos war aber das Zusammenspiel mit Simons Basic, für das ebenfalls ein kurzes Programm geladen werden mußte. Sogar der normale High-Resolution-Bildschirm für die Grafiken bleiben erhalten, wenn ein zweiter Monitor angeschaltet ist. Lediglich die Befehle TRACE, COLD, CENTRE und PRINT AT sind in ihrer Funktion eingeschränkt, beziehungsweise sollten nicht angewendet werden.

Leider gibt es bisher kein Textverarbeitungsprogramm, das mit der 80-Zeichenkarte zusammenarbeitet. Gerade hier wäre aber der sinnvollste Einsatz zu sehen. Hier sind die Softwarehäuser gefordert, denn Vizawrite 64 oder Textomat mit einer vollen Darstellung von 80 Zeichen auf dem Bildschirm wäre schon eine tolle Sache.

(Arnd Wängler/aa)

Bezugsquellen: Taxan Monitore, Melchers & Co., Postfach 102239, 2800 Bremen 1, Tel. (0421) 176989
BMC Monitore: Weber Computertechnik, Eulenspiegelstr. 56, 8000 München 38, Tel. (089) 6012564
Cable Monitore: Boston Computer, Rosenheimer Str. 145A, 8000 München 80, Tel. 491073
Decam Karte: Decam Electronic, Pappelweg 3, 7517 Waldbronn, Tel. (07243) 69264


Bild 7. Der Zwillingsbruder, der BMC BM 12 EN

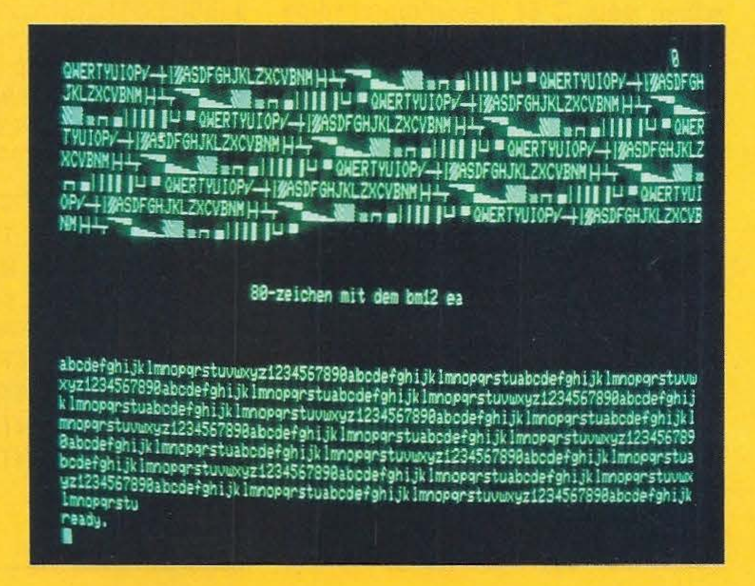


Bild 8. Dieser Monitor lieferte im Test das beste Ergebnis mit der 80-Zeichen-Karte. Leider ist das Foto etwas überstrahlt aufgenommen.

Print 64 - das universelle Interface



Bild 1. Print 64 — unscheinbar aber gut

Interface ist nicht gleich Interface. Jede Schnittstelle zum Anschluß von Druckern mit Centronics-Schnittstelle hat ihre eigenen Vor- und Nachteile. Ein Universalkönner ist Print 64.

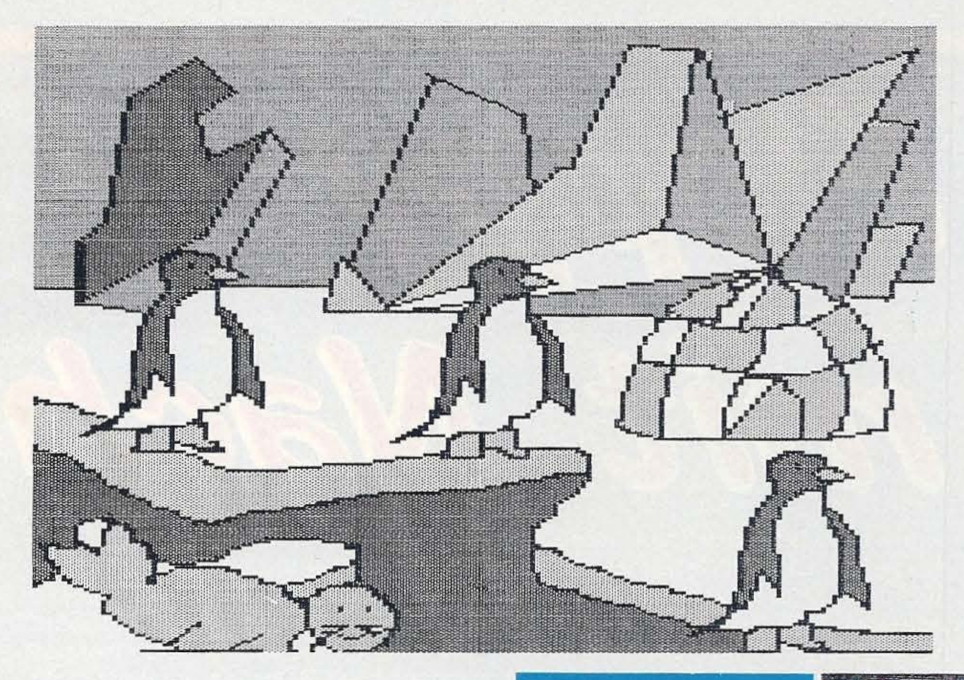
Manche Schnittstelle eignet sich gut zum Ausdruck von Texten, versagt aber bei den Commodore-eigenen Zeichen für Grafik und Cursorsteuerung. Andere wieder streiken bei den Grafiksymbolen und selbst definierten Zeichen. Nur ganz wenige beherrschen die Kunst des Bildschirmausdrucks hochauflösender Grafik. Für Print 64 gehören diese Funktionen aber zur Standardausstattung. Das Interface besteht aus einem unscheinbaren braunen Kasten (Bild 1), der zwischen dem C 64 (beziehungsweise Floppy) und dem Drucker zwischengeschaltet wird. Die notwendige Stromversorgung holt sich das Interface aus dem Drucker. Bei manchen Geräten (zum Beispiel Epson), ist es, bevor der Drucker sich rührt, notwendig, eine Stromversorgung von 5 V auf Pin 18 des Centronics-Ports zu legen. Der etwas Geübte entnimmt die Spannung einem der vielen TTL-Bausteinen auf der Druckerplatine und verbindet sie mit dem Port im Drucker selbst. Eine zweite Möglichkeit besteht darin, ein externes Netzteil anzuschließen.

Nach dem erfolgreichen Überwinden dieser Schwierigkeit steht dem Anwender ein leistungsfähiges Hilfsmittel zur Verfügung, das er bald nicht mehr missen möchte. Print 64 ist durch seine Konzeption als »Hardware-Schnittstelle« zu fast allen kommerziellen Programmen kompatibel. Bei der Wahl der Geräte und Sekundäradressen wurde darauf geachtet, möglichst dem Commodore-Standard zu entsprechen. Die Sekundäradresse 7 bewirkt beispielweise, wie bei Commodore-Druckern, die Groß- und Kleinschrift. Damit sind die Fähigkeiten des Interfaces natürlich noch bei weitem nicht erschöpft. Im wesentlichen kann man bei Print 64 zwischen zwei Bereichen von Funktionen unterscheiden. Zum einen sind das die Druckbefehle, die direkt auf den Drucker (über Sekundäradressen) wirken, zum anderen sind das einige Erweiterungen, die von der mitgelieferten Diskette nachgeladen werden.

Die Sekundäradressen finden Verwendung beim Ausdrucken von Texten und zum Auflisten von Programmen. Alle ASCII-Codes werden dabei so umgewandelt, daß sie der Standardtabelle entsprechen. Für besondere Zwecke kann das Interface auch als reines Verbindungskabel verwendet werden, ohne das eine Umwandlung der ASCII-Codes vorgenommen wird. Zusätzlich stehen aber auch fast alle Commodore-Grafikzeichen zur Verfügung. »Fast« alle Grafikzeichen muß deshalb gesagt werden, weil einige der Zeichen umdefiniert wurden und nun den deutschen Zei-

```
1 REM "£1■■■■■■DGSHE$♥-||■JDJDJ|||/\"
2 REM"■ ■ ■ ■ # @ £ "
10 REM L:LKJ:DFJFASFDFASFDFASFDFASDSDFAS
FDFASFDFASFDFASFSFDFASFDFDFASFDF
100 PRINT"■   LISTING MODUS "
110 PRINT
130 PRINT"   C64 AUF FX 80
140 PRINT:PRINT"   AMERIKANISCHER ZEICHE
NSATZ"
```

Bild 2. Jedes Listing wird sehr deutlich

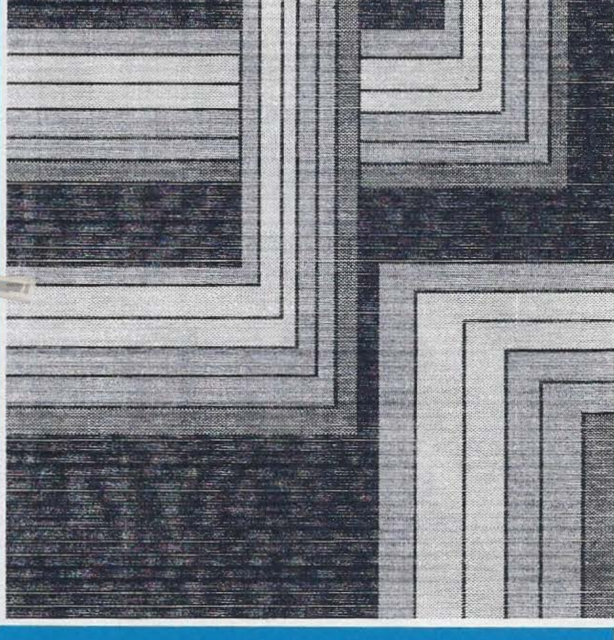


verschiedenen Graustufen für die Farbwerte des Bildes erreichen. Ein einziges Bild kann auf diese Weise in vielfältiger Weise variiert werden. Zum Ausdruck von eigenen Grafiken ist dieses Programm allerdings nicht unbedingt erforderlich. Dazu genügt schon das dritte Programm auf der Diskette. Es bereichert das Commodore-Basic um einige wichtige Befehle. Durch einfaches Ändern eines Befehls (beispielsweise !CTN oder !CTD) wird der Ausdruck sowohl in der Punktdichte als auch in der Breite beeinflußt. Es ist dabei gleichgültig, ob es sich um einen Textbildschirm oder um einen HiRes-Bildschirm handelt, es braucht nur ein anderer Befehl verwendet werden.

Bild 4. Sogar Bilder des »Koala Painters« sind kein Problem



Bild 3. Aus jeder Farbe wird eine Graustufe



chensatz beinhalten. So ist beispielsweise die eckige Klammer links das kleine »ä« und das Pfundzeichen das kleine »ö«. Erstmals hat der Commodore-Besitzer die Möglichkeit, in seinen Basic-Programmen, den deutschen Zeichensatz einzusetzen. Die Bildschirmdarstellung ist dann allerdings gewöhnungs- beziehungsweise umprogrammierungsbedürftig. Für Grafikzeichen benutzt das Interface den 8-Bit-Grafikmodus, wie er bei Epson-, Star- oder Seikonic-Druckern zur Verfügung steht, der Hintergrundspeicher bleibt deshalb zur eigenen Verwendung frei. Alle Zeichen werden bildschirmgetreu ausgedruckt, selbstverständlich auch invertiert. Dabei werden für jeden Modus zwei verschiedene Sekundäradressen eingesetzt. Die erste bewirkt, daß zur besseren Lesbarkeit alle Zeichen (auch die Cursorzeichen) mit einem kleinen Abstand gedruckt werden. Die zweite läßt alle Zeichen wie gewohnt aneinandergereiht. Beim Auflisten von Programmen ist diese Funktion besonders nützlich, denn die Deutlichkeit aller Steuerzeichen ist auffallend verbessert (Bild 2).

Der zweite interessante Bereich dieses voll grafikfähigen Interfaces bedarf des Nachladens einiger kurzer Programmteile von Diskette. Diese Programme sind reine Erweiterungen, wie sie eigentlich zu jedem Interface gehören sollten. Den großen Pluspunkt sammelt Print 64 durch seinen beigelegten Druckerkursus »Ein Interface stellt sich vor«, in dem alle wichtigen Funktionen ausführlich und grafisch ansprechend erläutert werden. Da der gesamte Druckerkursus auf Diskette gespeichert ist, können alle gerade gelernten Befehle sofort ausprobiert werden. Die beiden anderen Programme sind schon richtige Basic-Erweiterungen. Mit ihnen wird der Grafikausdruck fast schon zum Hobby. Mit einem speziellen Hardcopy-Programm können alle Bilder der bekannten Grafikhilfen Paint Magic, Supergrafik, Doodle und vor allem vom Koala-Painter und dem Sketch-Pad ausgedruckt werden (Bild 3 und 4). Die Bedienung ist menügesteuert und anwenderfreundlich.

Besonders reizvolle Effekte lassen sich aber durch Veränderungen der

Schenkt man dem Fehlen einiger weniger Grafikzeichen keine Beachtung, so sind der etwas langsame Ausdruck der Grafikzeichen und die Lötarbeiten beim Einbau in Epson-Drucker, die einzigen Nachteile dieses Interfaces. Print 64 arbeitet mit verschiedenen Textverarbeitungsprogrammen zusammen. Es kann deshalb und wegen seines insgesamt gelungenen Konzepts, als eines der besten Interfaces zum Anschluß von Druckern mit Centronics-Schnittstelle bezeichnet werden. Bei einem Preis von 320 Mark, einschließlich der Diskette mit den Hilfsprogrammen, eine starke Leistung.

(Arnd Wängler/aa)

Bezugsquelle: Rolf Rocke Computer, 5090 Leverkusen 3, Auestraße 1, Tel. 02171/2624

Floppy mit Nach

Es ist schon nervig. Wieder einmal sitzt man vor seinem C 64 und lädt ein Programm nach dem anderen in den Speicher. Vielleicht ordnen Sie gerade Ihre Programmsammlung. Diskette wechseln, Programm laden, kurz anschauen, danach wieder Diskette wechseln und abspeichern. Die Wartezeiten beim Laden und Speichern machen ganz müde. Die Finger trommeln einen ¾-Takt auf den Schreibtisch. Im Geiste rechnet man schon aus, wieviel wertvolle Zeit wieder verloren geht. Voller Neid denkt man an Disketten-Laufwerke anderer Hersteller und Computer, die die Nerven nicht so stark beanspruchen.

Ein erster Lichtblick

Eines Tages boten zwei junge Männer ein Programm an, das die Floppy 1541 beim Laden von Programmen um ein mehrfaches beschleunigt. Den Erfolg kennen Sie bereits. In der Ausgabe 10/84 wurde HYPRA-LOAD zum Listing des Monats gemacht.

Setzt man dieses Programm ein, werden sofort Sehnsüchte wach. Wenn schon das Laden von Programmen schneller geht, dann bitte auch das Speichern und wenn möglich auch Direktzugriffe und Dateihandling.

Es ist seltsam: Kaum hatten wir HYPRA-LOAD in den Händen, kam die nächste Überraschung. Ein Vertreter der holländischen Firma Computing International präsentierte uns ein System, daß alle oben genannten Wünsche in Erfüllung gehen ließ. Allerdings reicht eine reine Software-Lösung nicht mehr aus. Auch an der Hardware muß jetzt manipuliert werden (siehe Bild).

Der Grundgedanke ist klar. Es gibt eigentlich nur zwei Hemmschuhe für die Floppy. Zum einen ist es die serielle Datenübertragung, und zum anderen die umständlichen Routinen des Floppy-DOS (Disk-Operation-System). International Computing löst beide Probleme. Sie schrieben die Busroutinen um, machten sie schneller und gleichzeitig wurde ein paralleler Anschluß hergestellt und zwar interessanterweise ein IEEE-488- Bus. Das ist nicht unbedingt notwendig, aber ein Vorteil, wenn Sie auf größere Commodore-Peripherie zugreifen wollen. So schlossen wir versuchsweise die Commodore-Floppy SFD 1001 mit 1000 KByte Diskettenspeicherkapazität an. Es war eine wahre Wonne, daß Gerät in Aktion zu sehen. Auf die SFD 1001 stellten wir die 1541 und ließen die beiden zusammen arbeiten. Die Übertragung von einem Gerät zum anderen, ganz gleich in welcher Richtung, ging problemlos über die Bühne. Und die Geschwindigkeit war so verblüffend, daß wir zuerst an einen Fehler dachten, als die READY-Meldung so schnell wieder zu sehen war. Aber es gab keine Fehler.

Wie steht es mit der Kompatibilität? Diese Frage ist noch wichtiger als das Maß der Geschwindigkeitssteigerung. Denn davon hängt die Einsetzbarkeit ab. Es wäre schlimm gewesen, wenn man lediglich die parallele Übertragung gegen die serielle getauscht hätte. Jedoch stehen beide zur Verfügung. Mit einem kleinen Schalter läßt sich jederzeit die gewünschte Übertragungsart einstellen. Das ist in zwei Fällen wichtig. Einmal kann es sein, daß Programme sich doch nicht über das parallele Kabel laden lassen (das kann passieren, wenn diese Programme selber den seriellen Bus abfragen und verändern). Zum anderen muß auf einen am seriellen Port angeschlossenen Drucker zugegriffen werden können. Leider ging das nicht, wenn der Schalter auf parallel stand. Durch einfaches Umschalten ließ sich das Problem beseitigen. Wir konnten zwei Vorab-Versionen testen. In der einen Version mußte beim Umschalten ein Reset durchgeführt werden, in der anderen Version war das nicht nötig. Ein im Speicher befindliches Programm wird dann nicht gelöscht. Nach Aussage von Computing International werden die ausgelieferten Versionen einen Druckerbetrieb unabhängig von der eingestellten Übertragungungsart garantieren.

Bei den Programmen gab es keine Lade- oder Speicherprobleme, ausgenommen einiger Programme mit eigenen Busroutinen, die jedoch relativ selten sind. Wenn das Programm aktiv ist, kann kein Kassettenrecorder angeschlossen werden. Dieses Problem kann jedoch auch mit einem Schalter gelöst werden, der den vorher belegten Speicherbereich wieder freigibt.

Die RS232-Schnittstelle ist nach Angabe des Herstellers voll funktionstüchtig.

Die Tests ergaben bei diesem System eine zwei- bis vierfache Steigerung der Geschwindigkeit beim Laden und Speichern von Programmen sowie eine geringfügigere Steigerung um den Faktor 1,2 bis 2 beim Arbeiten mit Dateien.

Es gibt zwei Wege, seinen Commodore 64 umzurüsten. Sie können entweder einen Bausatz bestellen (zirka 250 Mark, inbegriffen sind alle Bauteile), oder Sie schicken Ihre Geräte (C 64 und Floppy 1541) ein, die Lösung für den Nichtbastler. Dann kostet der Umbau voraussichtlich 375 Mark. Den Umbau erledigt in Deutschland die Firma Optronik Service International.

Speeddos — die Erweiterung mit Pfiff

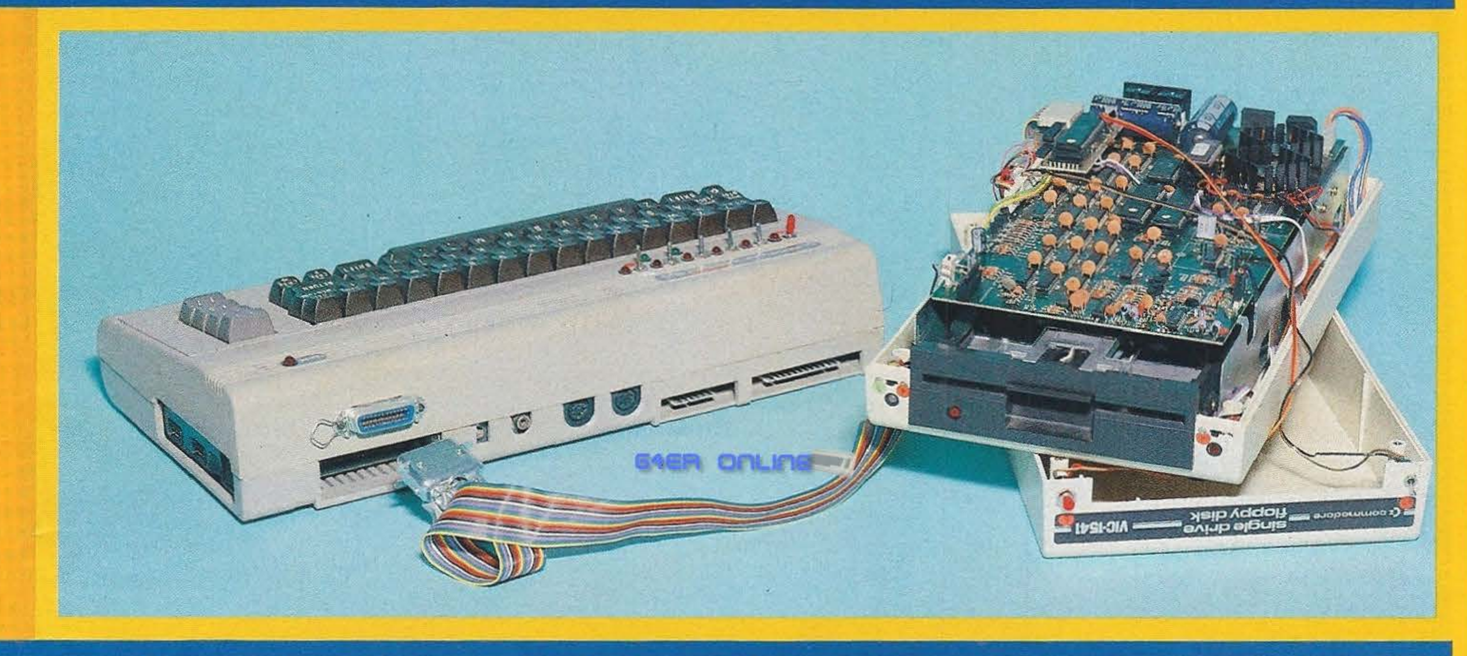
Eine weitere Alternative zu diversen Fastload-Programmen und IEC-Bus-Floppy-Laufwerken vertreibt S&S-Soft und nennt sich Speeddos: Eine kleine Hardware-Erweiterung für den C 64, deren Einbau sich auf einfaches Einstecken beschränkt.

Alle Programme, die mit »Speeddos« abgespeichert wurden, sollen sich bis zu zehnmal schneller laden

Schimpfen Sie nicht mehr über die nervenzermürbende Langweilerin Commodore-Floppy 1541. Jetzt kommen auf einmal gleich mehrere Systeme auf den Markt, die der 1541 die Sporen geben. Die besten drei stellen wir Ihnen vor.

brenner

Bild 1. Eines der vorgestellten Systeme. Bei diesem System wurde ein IEEE-488-Bus integriert. Dabei bleiben Expansion- und User-Port frei. An Hardware sind notwendig je eine kleine Platine für C 64 und Floppy, außerdem ein IEEE-Stecker plus Kabel. Die seriellen Ports bleiben voll funktionstüchtig. Die Platinen werden aufgesteckt.



lassen. Mit normalem Betriebssystem gespeicherte Programme werden mit rund sechsfacher Geschwindigkeit geladen.

Da die reine Datenübertragung etwa achtmal schneller ist, ergibt sich bei floppyinternen Operationen eine generelle Geschwindigkeitssteigerung um den Faktor 2 bis 5. Das Formatieren einer Disk benötigte in unserer Demonstrationsversion 39 Sekunden, in der Verkaufsversion soll der Formatierungsvorgang nur noch 17 Sekunden dauern und das Laden soll mit der zehnfachen Geschwindigkeit möglich sein. Auch der Betrieb von zwei 1541 mit Speeddos ist kein Problem.

Zusätzlich zum Geschwindigkeitsvorteil wurden einige Verbesserungen und Erweiterungen an den Betriebssystemen von Computer und Floppy eingebaut. Dem Benutzer steht ein komplettes DOS zur Verfügung, mit dessen Hilfe man auch den zusätzlichen Löschschutzbefehl der Floppy ansprechen kann oder das Inhaltsverzeichnis ohne Programmverlust erhält. Die Funktionstasten wurden mit einigen häufig benötigten Funktionen belegt (zum Beispiel List, Run, Directory ohne Programmverlust, etc.). Die Belegung ist abschaltbar. Außerdem wurde eine softwaremäßige Centronics-Schnittstelle integriert, die automatisch aktiviert wird, wenn ein Centronics-Drucker (zum Beispiel Itoh oder Epson) am User-Port angeschlossen ist. Entsprechende Kabel dazu können ebenfalls geliefert werden.

In der Verkaufsversion soll außerdem ein komfortabler Maschinensprache-Monitor (mit Auf- und Abwärts-Scrolling) enthalten sein.

Darüber hinaus bietet Speeddos noch einen laufwerks- und diskettenschonenden Effekt: Bei auftretenden Fehlern hört man kein lautes Rattern mehr, sondern nur noch ein leises »Taktak«, das den Steppermotor und den Endanschlag nur noch minimal belastet. Beim Einlegen von Disks in das Floppy-Laufwerk läuft der Motor automatisch an und zentriert die Diskette gleich beim Einlegen.

Speeddos benötigt keinen Speicherplatz und läuft mit allen Programmen, die nicht auf den Datasette-Recorder oder die RS232-Schnittstelle zugreifen. Das sind rund 98% der derzeit verfügbaren Software. Somit ist aber auch hier zum Beispiel die Verwendung eines Akustikkopplers in Verbindung mit Speeddos nicht möglich. In diesem Fall muß das EPROM vom C 64 abgezogen werden. Der Betrieb mit CP/M läuft da hingegen ohne Einschränkung und geht ungefähr dreimal schneller als normal.

Besonders hervorzuheben an dieser Erweiterung ist die außerordentliche Geschwindigkeit und Kompatibilität zur erhältlichen Software.

Speeddos wird als Bausatz voraussichtlich 269 Mark kosten. Der Einbau beschränkt sich auf einfaches Einstecken von EPROMs und

Floppy mit Nachbrenner

Steckern in den Computer und ist von jedem Laien durchführbar. Für diejenigen, die einen C 64 mit eingelötetem Betriebssystem besitzen, ist gegen einen Aufpreis von 20 Mark eine Platine erhältlich, die in diesem Fall das Löten erspart.

Die Entwickler von Speeddos, O. Joppich und O. Eikemeier, arbeiten an einem Backup-Programm für Speeddos, das bei einem Laufwerk eine ganze Disk in unter zwei Minuten kopieren soll. Ein noch schnelleres Backup für zwei Laufwerke ist ebenfalls in Planung.

Geschwindigkeit ist (k)eine Hexerei

Auch mit dem dritten System, das wir Ihnen vorstellen, dem »Turbo Access« von Roßmüller, werden Träume wahr: Die »lahme Ente« — gemeint ist natürlich die Floppy 1541 — lernt fliegen. Zehnmal schnelleres Laden, dreimal schnelleres Abspeichern, das sind Werte, die sich sehen lassen können. Ja selbst die großen Commodore-Floppies mit ihrem Parallelbus können auf einmal nicht mehr mithalten.

Die Engstelle der 1541 ist seit langem bekannt: Alle großen CBM-Computer benutzen einen 8-Bit-Parallelbus, nur die kleinen arbeiten mit einer seriellen Schnittstelle. Dabei wird jedes übertragene Byte in 8 Bit zerlegt, die einzeln über den Bus geschickt werden. Und das braucht Zeit ...

Acht Drähte machen noch keinen Bus ...

Was liegt also näher, als einfach mehr Leitungen zur Übertragung zu spendieren? Doch damit ist das Betriebssystem der Floppy, das DOS, nicht so ohne weiteres einverstanden. Man muß es daher umgehen und eine eigene Verwaltung aufbauen. Davon machen Programme wie »HYPRA LOAD« gebrauch: In das RAM der Floppy wird ein eigenes Programm geschrieben, das die Datenübertragung abwickelt. Der Nachteil einer solchen Software-Lösung liegt auf der Hand: Vor jeder Übertragung muß das Programm erst in die Floppy geschoben werden, und dazu braucht man wieder ein Programm, das geladen werden muß ...

... ohne Lenkung geht nichts!

Beim »Turbo Access« hat man daraus die Konsequenz gezogen: Es handelt sich um eine reine Hardware-Lösung, die aus drei Platinen und einem Flachbandkabel für die Parallelübertragung besteht.

Zur Ausstattung gehören zwei EPROMs, die neue Betriebssysteme sowohl für die Floppy, als auch für den C 64 enthalten. Sie befinden sich zusammen mit den alten umschaltbar in Floppy beziehungsweise Computer. Die dritte Platine steckt im Expansion-Port und ist mit einem zusätzlichen PIA, ähnlich dem User-Port, bestückt. Für Interessierte sei hier angemerkt, daß sie auf keinen Port verzichten müssen — der benutzte Expansion-Port ist durchgeschleift.

Allerdings lag zum Test noch nicht die endgültige Fassung, sondern eine vorläufige »Arbeitsversion« vor. Nach Angaben des Herstellers wird die endgültige Version auch berücksichtigen, daß die Floppy 1541 in zwei Versionen gebaut wird, die sich in der Platinenform unterscheiden.

Wer jetzt Holz aufs Feuer legt, um den Lötkolben aufzuheizen, muß allerdings enttäuscht werden. Alles ist steckbar, mit wenigen Griffen einzubauen.

Was leistet »Turbo Access«

Stellt man die Floppy wie oben beschrieben auf Parallelübertragung um, erreicht man aus dem Stand die 6fache Ladegeschwindigkeit. Noch sind die Files ja nach alter DOS-Manier abgelegt. Dies ist mittlerweile schon fast »Standard«. Mit »Turbo Access« abgespeicherte Programme werden allerdings zirka zehnmal schneller geladen. Hier einige Testzeiten:

158 Blöcke werden in 10,4 Sekunden geladen. Die 4040 braucht dafür immerhin 23 Sekunden, die »normale« 1541 1 Minute und 40 Sekunden.

Eine Diskette wird in 29 Sekunden formatiert. Die 1541 braucht etwa 1 Minute 30 Sekunden.

Die Befehle SCRATCH und VALIDATE werden zirka doppelt so schnell wie bisher ausgeführt.

SAVE erfolgt mit der dreifachen Geschwindigkeit. Das entspricht ungefähr der Geschwindigkeit der 4040. Bei jedem SAVE wird anschließend verifiziert. Daraus erklärt sich der erhöhte Zeitbedarf gegenüber LOAD.

Wodurch werden diese Zeiten möglich? Ursache ist, wie bei den beiden anderen Systemen, daß nicht nur die Übertragung selbst, sondern das gesamte DOS umfassend überarbeitet wurde. Dabei wurde auch die Verwaltung auf der Diskette abgeändert, und das bringt die Floppy ganz schön auf Trab. Die Kopfpositionierung wird dreimal schneller, das zahlt sich vor allem bei relativen Files aus. Und einige Fehler im alten DOS wurden gleich mitkorrigiert. Das lästige »Gerappel« beim Formatieren, Ursache für viele verstellte Schreib-Lese-Köpfe, wird auf ein Mindestmaß reduziert und noch einiges andere mehr.

Keine Angst, bei allen Änderungen wurde darauf geachtet, daß alle Befehle genauso funktionieren wie bisher. Denn was nützt die schnellste Floppy, wenn kein Programm damit zusammenarbeitet. Bei allen Versuchen gab es bisher nur ein Programm, das nicht mit »Turbo Access« zusammen lief. Selbst Programme, die intensiv mit Diskettenbefehlen arbeiten (zum Beispiel »EX-DOS« und Kopier-Programme wie »FCOPY« oder »Quick-Copy«), machten ebensowenig Schwierigkeiten wie Datenverwaltungs- oder Testprogramme. Spiele, die häufig Teile von Diskette nachladen, werden zum echten Streß, da man nicht einmal mehr dazu kommt, Bier aus dem Keller zu holen ...

Relative oder sequentielle Files werden genauso problemlos verarbeitet. Und sollte einmal etwas nicht laufen, drückt man CTRL + £, legt einen Schalter um und hat wieder den alten Zustand hergestellt. In den meisten Fällen ist das sogar mitten im Programm ohne Programm- oder Datenverlust möglich.

Und der Haken bei der Sache?

Natürlich, das dicke Ende mußte ja noch kommen, aber so dick ist es nun auch wieder nicht. Um im Rechner-Betriebssystem Platz zu bekommen, wurden die Kassetten-Routinen entfernt. Für den Betrieb der Datasette muß man auf die alten Betriebssysteme umschalten. Dafür wurden im neuen Betriebssystem noch einige äußerst nützliche Routinen eingebaut. So ist jetzt mit CTRL + D jederzeit eine Directory-Ausgabe — natürlich ohne Programm-

Fortsetzung auf Seite 163



Nichts ist ewig

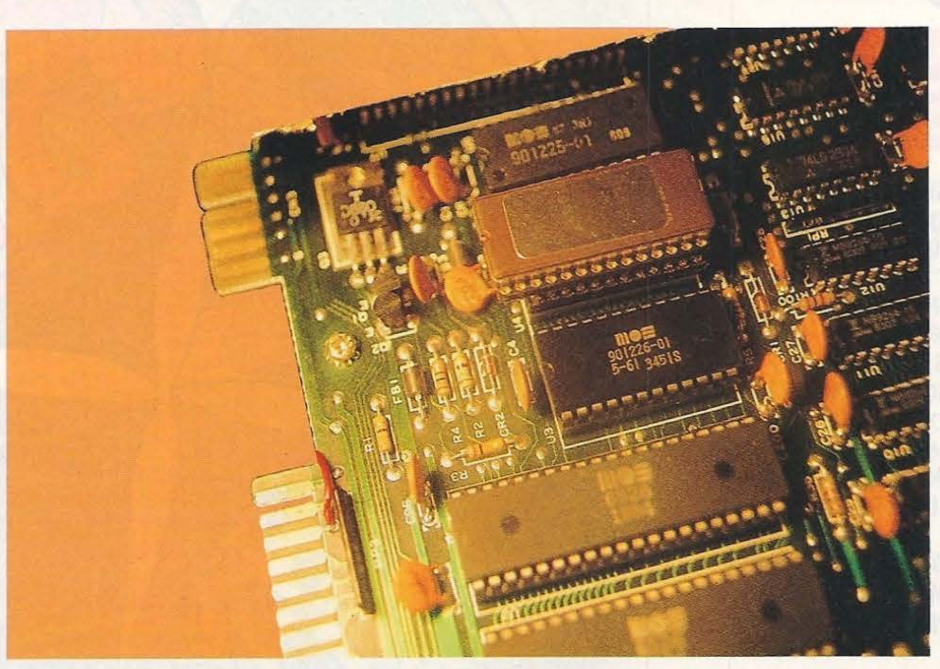


Bild 4. Das EPROM 2754 zweifach »gesockelt«

Zugegeben, der Commodore 64 hat einige Nachteile. Aber warum sollte man sich damit abfinden? Alles kann mit dem ROM-Change-Programm verändert werden.

Jedem Computer, auch dem Commodore 64 wird ab Werk eine bestimmte Ausstattung an Software mit auf den Weg gegeben. Damit ist nicht die Demo-Diskette gemeint, sondern die fest im Computer eingebaute, auf PROMs gespeicherte Firmware. Sie sorgt dafür, daß der Commodore überhaupt auf Eingaben reagiert oder einen Basic-Befehl ausführt.

Der gesamte C 64 (und jeder andere Computer) ist eigentlich nichts anderes als eine Anzahl miteinander verdrahteter Baugruppen, die allein zu nichts fähig sind. Leben eingehaucht wird dem ganzen erst durch ein sehr wichtiges Programm, das sogenannte Betriebssystem. Dieses Programm initialisiert und verwaltet die gesamte Hardware. Beim Commodore 64 ist es genau 8 KByte lang und liegt im Bereich von $E000 bis $FFFF. Der zweite wichtige Festwertspeicher ist der Basic-Interpreter. Er ist ein Übersetzungsprogramm, das eine Anweisung in ein maschinengerechtes Signal umwandelt. Auch der Interpreter benötigt 8 KByte und liegt im Bereich von $A000 bis $C000. Jetzt fehlt nur noch der Charakter-Set von 4 KByte, der ebenfalls in einem eigenen ROM untergebracht ist.

Diese drei Programme sind für das äußere Erscheinungsbild und die Funktionalität des Commodore 64 verantwortlich. Hier eröffnet sich ein extrem interessanter Bereich der Programmierung. Dazu bieten sich zwei Wege an: Der erste Weg beruht auf der glücklichen Tatsache, daß es beim Commodore möglich ist, den RAM-Bereich unter einem ROM zu nutzen. In der Praxis sieht das folgendermaßen aus: Die Speicherinhalte des ROMs werden zum Bearbeiten in den darunter liegenden RAM-Bereich kopiert. Ob nun das RAM oder das ROM aktiv ist, entscheidet das 6510 Datenrichtungsregister (Speicherstelle 1). Vom Basic aus ist dieses Register allerdings nur dann zu verändern, wenn sowohl der Basic-Interpreter, als auch das Betriebssystem in das RAM kopiert wurden. Das geschieht entweder mit einer POKE-Schleife oder mit dem ROM-Change Programm. (Betriebssystem kopieren: FOR A=57344 TO 65535: POKE A,PEEK (A) : NEXT)

(Basic kopieren: FOR A=40960 TO 49152: POKE A, PEEK (A) : NEXT). Der Normalwert dieses Registers ist 55 (probieren Sie es aus). Soll das RAM (für Basic) selektiert aktiv sein, muß hier der Wert 54 eingeschrieben werden. Für Basic und Betriebssystem zusammen beträgt der Wert 53. Alle Veränderungen des Basic und Betriebssystems sind dann aktiv. Bekannte Programme wie Quickcopy und viele Basic-Erweiterungen funktionieren nach diesem Prinzip.

Der zweite und wesentlich reizvollere Weg die Firmware zu beeinflussen ist, das Betriebssytem dauerhaft zu verändern. Dazu ist aber ein Eingriff im Computer notwendig, denn die oben beschriebenen Bausteine müssen gegen andere ausgetauscht werden. Wer also noch Garantie auf seinen Commodore hat, sollte besonders vorsichtig sein. Im ersten Teil dieses Artikels wollen wir, zum Einüben sozusagen, die Funktionstasten des Commodore 64 mit bestimmten, oft gebrauchten Werten belegen. Im zweiten Teil wird das

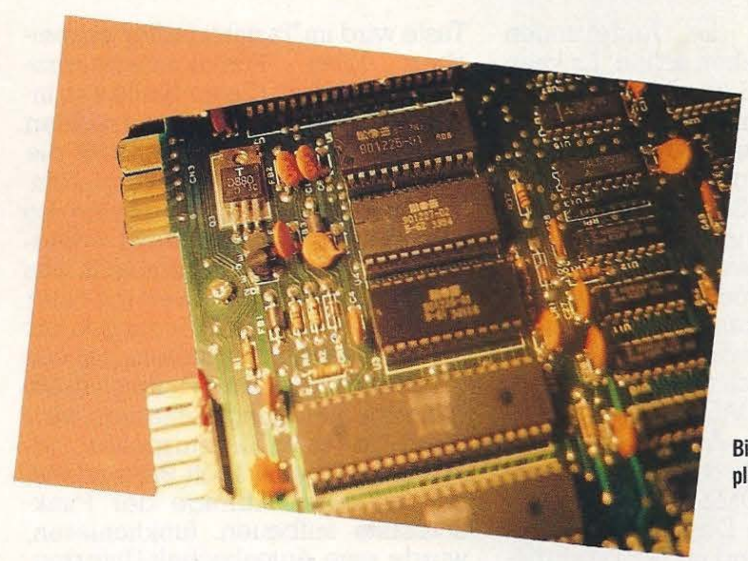


Bild 1. Die Steckplätze U3 bis U5

2764		2364	
1 ○	28		
2	27		
3	26	1 ○	24
4	25	2	23
5	24	3	22
6	23	4	21
7	22	5	20
8	21	6	19
9	20	7	18
10	19	8	17
11	18	9	16
12	17	10	15
13	16	11	14
14	15	12	13

Bild 2. Der 2764 (links) und der 2364 (rechts)

Bild 3. So wird der 2764 mit den Pins des 2364-Sockels verbunden

2764		2364
1,28,27,26	→	24
2	→	21
3	→	1
4	→	2
5	→	3
6	→	4
7	→	5
8	→	6
9	→	7
10	→	8
11	→	9
12	→	10
13	→	11
14	→	12,20
15	→	13
16	→	14
17	→	15
18	→	16
19	→	17
20,14	→	nicht angeschl.
21	→	19
22	→	20
23	→	18
24	→	22
25	→	23
1,28,27,26	→	24

Hypra-Load aus der Ausgabe 10/84 im Betriebssytem verankert. Der Nachteil dieser Änderungen soll nicht verschwiegen werden. Da alle neuen Funktionen natürlich Speicherplatz benötigen, müssen andere Teile des Betriebssystems entfallen. Wir haben uns entschlossen, die Kassettenroutinen ab $F800 herauszunehmen und zu überschreiben. Das Laden von Kassette wird dadurch unmöglich, es sei denn, das alte Betriebssystem wird parallelgeschaltet.

Bevor wir nun auf das Hilfsprogramm für diese Veränderungen, das ROM-Change-Programm, eingehen, sollen die Hardware-Voraussetzungen für die Änderung des Betriebssystems erklärt werden. Nach dem Öffnen des Computers finden wir auf der linken hinteren Seite drei kleine Bausteine, hinter denen auf der Platine die Bezeichnungen U3 bis U5 stehen. U3 ist der Basic-Interpreter, U4 das Betriebssytem, U5 das Charakter-ROM (Bild 1). Heute interessieren wir uns aber nur für den U4-Steckplatz. Wer Glück hat, findet dort einen Stecksockel. Wer Pech hat, muß seinen Händler oder einen Lötkolbenfachmann bitten, ihm hier einen Stecksockel einzulöten. Anstelle des dort befindlichen ROMs kann aber, und das ist die wesentlichste Veränderung, auch ein EPROM stecken. Am besten eignen sich hierzu die 2564-Typen, denn sie sind pinkompatibel zu den Commodore-ICs. Leider sind sie sehr schwer zu beschaffen. Im Normalfall wird aber wahrscheinlich ein 2764-EPROM Verwendung finden. Der Nachteil dieses EPROMs besteht in einer anderen Belegung der Anschlußpins. Hier hilft allerdings ein einfacher Adaptersockel. Dazu braucht man einen 24- und einen 28-Pin-Stecksockel. Diese beiden Sockel werden miteinander verdrahtet (Bild 2 und 3). Bild 2 zeigt die beiden Sockel mit den Beinen nach unten stehend. Der kleine Sockel steckt später im U4-Steckplatz, der größere Sockel trägt das geänderte Betriebssystem und steckt auf dem kleineren Sockel. Vor dem Einbau ist es aber ratsam, alle Kontakte auf richtigen Anschluß und Leitfähigkeit zu überprüfen. Schon ein einziger falscher Anschluß führt zum »Absturz« des gesamten Systems. Bild 4 zeigt, wie der neue Sockel mit der EPROM-Kerbe zur Gehäuse-Rückseite auf der Platine steht.

Nun aber zur Praxis, dem ROM-Change-Programm. Es ermöglicht das gefahrlose Ändern und Ausprobieren aller Umprogrammierungen. Dazu wird nach dem Start der selektierte ROM-Bereich (Betriebssytem oder Basic) ab Adresse $6000 ins RAM kopiert. Das geschieht mit einem kurzen Maschinenprogramm, das im Kassettenpuffer steht. Das Programm ist für den Betrieb mit einem Diskettenlaufwerk gedacht, läuft aber auch im Kassettenbetrieb, wenn ein eigener Monitor zum Laden und Abspeichern Verwendung findet. Nach dem Kopieren erscheint das Hauptmenü, von dem aus alle wichtigen Funktionen erreichbar sind. Die erste dient dem Einlesen von fest im Programm eingebauten DATA-Zeilen. In unserem Beispiel sind ab Zeilennummer 8000 die DATAs für die Funktionstasten einprogrammiert. Hier können natürlich auch eigene Werte stehen. Die erste Zahl gibt dabei die reelle Adresse an, ab der die DATAs geschrieben werden sollen (zum Beispiel 57612 = $E10C). Der Computer errechnet dann die entsprechende Stelle im RAM. Die zweite Zahl gibt an, wieviel Bytes übertragen werden sollen. Die dritte Zahl ist die Prüfsumme. Danach folgen die Programm-DATAs. Bei einem Prüfsummenfehler zeigt der Computer die falsche Prüfsumme an. Eigene Daten werden einfach an die vorhandenen DATAs angehängt. Die ersten drei Bytes müssen natürlich auch die obige Bedeutung haben. Als letzte Zeile muß stehen: DATA 0, da es sonst zu einem OUT OF DATA ERROR kommt. Der zweite Menüpunkt liest Maschinenprogramme direkt an die vorgesehene Stelle. Damit kann man beispielsweise ein Programmfile, das von einem Assembler erzeugt wurde, direkt einlesen. Wichtig ist, daß die Startadresse des Programms im ROM-Adreßbereich beziehungsweise im Bereich $6000 liegt. Der mit Punkt 3 wählbare Minimonitor hilft beim schnellen Ändern einzelner Bytes

im hexadezimalen Zahlensystem. Die Startadresse entspricht dabei der Adresse im ROM. Der Minimonitor wird durch Eingabe einer Zahl größer $FF oder des Buchstabens X verlassen. Will man keine Speicherstelle ändern, genügt RETURN für die Übernahme des alten Wertes. Für größere Änderungen reicht dieser Minimonitor natürlich nicht mehr aus. Dazu wird mit Punkt 4 ein eigener Monitor aktiviert. Dieser muß allerdings vor dem Start des ROM-Change-Programms bereits geladen sein. Der Monitor darf im Bereich von $8000 bis $9FFF oder von $C000 bis $CFFF stehen. Zum Starten genügt das Eingeben der Startadresse des Monitors (hexadezimal!). Mit den Menüpunkten 5 und 6 wird der Speicherbereich von $6000 bis $7FFF geladen beziehungsweise abgespeichert. Das Laden eines kompletten Betriebssystems dauert allerdings mehrere Minuten. Einer der wichtigsten Menüpunkte ist aber der siebte. Er startet das gerade veränderte oder geladene neue Betriebssystem (es wird natürlich zuerst in seinen richtigen Speicherbereich verschoben). Vor dem Ausprobieren dieser Funktion muß in jedem Fall der Punkt 5 angewählt werden, wenn die Änderungen nicht verlorengehen sollen. Es kann vorkommen, daß das Betriebssystem beim Starten »abstürzt«. Dann genügt ein RESET und die erneute Aktivierung des geänderten ROMs durch POKE 1,53.

Sind alle Tests im RAM positiv verlaufen, können wir uns an die Speicherung des neuen Betriebssytems auf EPROMs heranwagen. Das neue Betriebssystem wird von Diskette in den RAM-Bereich ab $6000 absolut, das heißt mit LOAD"Ihr Betriebssystem",8,1 geladen. Ab dieser Adresse beginnt auch die Programmierung des EPROMs. Sie endet bei $8000 (8 KByte). Das neue EPROM wird anschließend in den beschriebenen Stecksockel eingesetzt und in Steckplatz U4 befestigt. Fertig ist das Betriebssytem.

Was noch zu klären bleibt, ist die im ROM-Change-Programm bereits eingebaute Funktionstastenbelegung. Die Abfrage der Tastatur geschieht im Betriebssytem über den Interrupt. Wird eine Taste gedrückt, hält der Prozessor das laufende Programm an und springt in die Interrupt-Unterroutine bei $EA31. Dort wird festgestellt, welche Taste gedrückt wurde. Der ASCII-Code der Taste wird im Tastaturpuffer gespeichert. Das Funktionstastenprogramm greift an dieser Stelle ein, indem es den Wert der gedrückten Tasten mit den ASCII-Codes für die Funktionstasten (133-140) vergleicht. Stimmt der Wert nicht überein, so wird das Programm normal weitergeführt. Ansonsten vergleicht das Programm den Tastenwert mit einer Tabelle, in der die Belegungen für die Funktionstasten stehen. Hat das Programm die zur Funktionstaste gehörige Belegung gefunden, wird diese in den Tastaturpuffer geschrieben. Damit auch Programme, die auf einer Abfrage der Funktionstaste aufbauen, funktionieren, wurde eine Autoabschalt-Unterroutine mitprogrammiert. Kommt es dennoch einmal zu Schwierigkeiten, werden die Funktionstasten mit POKE 2,1 abgeschaltet und mit POKE 2,0 wieder aktiviert.

Abschließend sei nochmals darauf hingewiesen, daß alle Arbeiten an der Hardware des C 64 mit einem nicht geringen Zerstörungsrisiko verbunden sind. Wer also im Umgang mit Lötkolben und EPROMs nicht geübt ist, sollte sich an einen Fachmann wenden.

(Arnd Wängler/Ernst Schöberl/aa)

Beim Abtippen des Programms sind die Werte in Klammern am Ende einer Zeile nicht miteinzugeben. Sie sind erst für eine spätere Ausgabe von Bedeutung. Unterstrichene Buchstaben sind mit der Shift-Taste, überstrichene mit der Commodore-Taste einzugeben. Bei Ausdrücken in eckigen Klammern ist die jeweilige Taste zu drücken.

```
100 REM SPEICHERPLATZ FUER ROM VOR BASIC SCHUET
   ZEN                                        <003>
110 POKE 55,0:POKE 56,96:CLR                  <120>
120 POKE 53272,23:POKE 53280,2                <212>
130 GOSUB 1130                                <212>
140 IF PEEK(53247)<>33 THEN 160               <055>
150 IF PEEK(53246)=0 THEN GOSUB 3600:GOTO 500
                    <111>
155 GOSUB 3700:GOTO 500                       <074>
160 PRINT"[DOWN2]":PRINT"[SPACE]DRUECKEN[SPACE]
   SIE":PRINT                                 <059>
170 PRINT:PRINT TAB(7);"1[SPACE2]UM[SPACE]
   KERNAL[SPACE]($E000-$FFFF)"                <219>
180 PRINT:PRINT TAB(7);"2[SPACE2]UM[SPACE]BASIC
   [SPACE]($A000-$BFFF)"                      <131>
190 PRINT:PRINT:PRINT TAB(11);"IN[SPACE]RAM
   [SPACE]AB[SPACE]$6000[SPACE]ZU[SPACE]
   KOPIEREN."                                 <072>
200 GET A$:IF A$=""THEN 200                   <039>
210 IF A$="1"THEN POKE 32767,255:GOSUB 3600
   :GOTO 240                                  <225>
220 IF A$<>"2"THEN 200                        <223>
230 GOSUB 3700                                <061>
235 REM AN=ANFANGSADRESSE ROM                 .
    EN=ENDADRESSE                             <231>
240 GOSUB 1600                                <068>
300 POKE K+8,INT(AN/256)                      <252>
310 POKE K+29,INT(EN/256)                     <061>
340 SYS K :REM SCHIEBEROUTINE                 <032>
500 POKE 53280,5:GOSUB 1130:PRINT:POKE 53247,0
                 <128>
505 PRINT:PRINT:PRINT"[SPACE6]ERWEITERUNG[SPACE]
   DES[SPACE]ROMS[SPACE]DURCH":PRINT          <052>
506 PRINT TAB(8);"1:[SPACE]DATA[SPACE]ZEILEN
   [SPACE]EINLESEN"                           <182>
510 PRINT TAB(8);"2:[SPACE]PROGRAMM[SPACE]VON
   [SPACE]DISK"                               <010>
520 PRINT TAB(8);"3:[SPACE]HEXMONITOR"        <032>
530 PRINT TAB(8);"4:[SPACE]EIGENEN[SPACE]
   MONITOR"                                   <193>
540 PRINT TAB(8);"5:[SPACE]ABSPEICHERN[SPACE]
   DES[SPACE]NEUEN[SPACE]ROMS"                <101>
545 PRINT TAB(8);"6:[SPACE]LADEN[SPACE]VON
   [SPACE]GEAENDERTEM[SPACE]ROM"              <021>
546 PRINT TAB(8);"7:[SPACE]STARTEN[SPACE]DES
   [SPACE]NEUEN[SPACE]ROMS"                   <106>
547 PRINT TAB(8);"8:[SPACE]PROGRAMM[SPACE]
   BEENDEN"                                   <137>
550 PRINT:PRINT:PRINT TAB(10);"BITTE[SPACE]
   WAEHLEN[SPACE]SIE!"                        <017>
560 GET A$:IF A$=""THEN 560                   <154>
570 I=VAL(A$):IF I<1 OR I>8 THEN 560          <200>
575 POKE 53247,33:POKE 53280,2                <160>
580 ON I GOTO 1000,2000,3000,4000,5000,6000,
   2500,3500                                  <001>
999 REM DATA ZEILEN EINLESEN                  <173>
1000 PRINT"[CLEAR]":PRINT:PRINT TAB(10);"LESEN
   [SPACE]DER[SPACE]DATA-ZEILEN":PRINT:PRINT
                  <051>
1010 GOSUB 1600                               <073>
1020 READ A:IF A=0 THEN 500                   <235>
1030 READ B:REM ANZAHL DER BYTES              <188>
1032 READ P1:REM PRUEFSUMME                   <230>
1033 P2=0                                     <113>
1035 D=A:GOSUB 1300:PRINT"$";A$;"-";:D=A+B-1
   :GOSUB 1300:PRINT"$";A$;                   <109>
1040 FOR I=A-OF TO A-OF-1+B                   <254>
1050 READ D:POKE I,D                          <115>
1051 P2=P2+D:IF P2>65535 THEN P2=P2-65536     <212>
1053 NEXT I                                   <236>
1055 IF P2<>P1 THEN 1070                      <132>
1057 PRINT"[SPACE3]OK"                        <156>
1060 GOTO 1020                                <116>
1070 PRINT"PRUEFSUMME[SPACE]FALSCH:[SPACE]";P2
                  <192>
1080 GET A$:IF A$=""THEN 1080                 <210>
1090 GOTO 1020                                <146>
```

Listing »ROM-Change«

```
1100 STOP                                          <224>
1130 PRINT"[CLEAR]":PRINT TAB(8);"PROGRAMM
   [SPACE]ZUM[SPACE]AENDERN[SPACE]DES":PRINT
                         <065>
1140 PRINT TAB(12)"BETRIEBSSYSTEMS":PRINT:PRINT
                         <050>
1150 PRINT"[SPACE]GESCHRIEBEN[SPACE]VON[SPACE]
   ERNST[SPACE]SCHOEBERL[SPACE](1984)"          <187>
1160 RETURN                                        <026>
1199 REM HEX IN DEZ UMWANDLUNG                     <163>
1200 D=0:FOR J1=0 TO LEN(A$)-1                     <235>
1210 B=ASC(RIGHT$(A$,1)):A$=LEFT$(A$,LEN(A$)-1)
                         <245>
1220 B=B-48:IF B>9 THEN B=B-7                      <197>
1230 IF B<0 OR B>16 THEN 1280                      <254>
1240 D=D+B*16↑J1                                   <0N2>
1250 NEXT J1                                       <227>
1260 RETURN                                        <126>
1280 D=-1:RETURN                                   <159>
1299 REM DEZ IN HEX                                <006>
1300 J2=INT(LOG(D)/LOG(16)):A$=""                  <084>
1310 FOR J1=J2 TO 0 STEP-1                         <166>
1320 K1=INT(D/16↑J1):D=D-K1*16↑J1                  <185>
1330 IF K1>9 THEN K1=K1+7                          <090>
1340 K1=K1+48:A$=A$+CHR$(K1)                       <245>
1350 NEXT J1                                       <072>
1360 RETURN                                        <227>
1600 RESTORE:K=828:FOR I=K TO K+32:READ A
   :POKE I,A:NEXT I                                <056>
1610 DATA 120,160,0,132,251,132,253,169,224,133,
   252,169,96,133,254,177,251,145                  <241>
1620 DATA 253,200,208,249,230,252,230,254,165,
   252,201,0,208,239,96                            <167>
1630 RETURN                                        <242>
1999 REM PROGRAMM VON DISK IN ROM EINFUEGEN
                         <003>
2000 GOSUB 1130:PRINT                              <252>
2010 PRINT TAB(2);"EINFUEGEN[SPACE]EINES[SPACE]
   PROGRAMMS[SPACE]VON[SPACE]DISK"                 <087>
2020 PRINT:PRINT:PRINT:INPUT"FILENAME:";A$ <200>
2030 OPEN 1,8,0,A$                                 <022>
2033 GOSUB 6100:I=ASC(A$):GOSUB 6100
   :I=I+256*ASC(A$):IF ST>0 THEN 6200              <052>
2035 PRINT:PRINT"FOUND[SPACE]";A$                  <198>
2036 IF I<AN THEN 2110                             <124>
2040 GET#1,A$:A$=A$+CHR$(0)                        <045>
2050 IF ST>0 THEN 2100                             <135>
2060 POKE I-OF,ASC(A$):I=I+1:GOTO 2040             <190>
2100 CLOSE 1:GOTO 500                              <101>
2109 REM PROGRAMM NICHT IN ROMBEREICH              <038>
2110 PRINT"ROM[SPACE]VON[SPACE]$";:D=AN
   :GOSUB 1300:PRINT A$;"-";:D=AN+8191
   :GOSUB 1300:PRINT A$                            <075>
2120 PRINT"STARTADRESSE[SPACE]DES[SPACE]
   PROGRAMMS[SPACE]BEI[SPACE]$";:D=I:GOSUB 1300
   :PRINT A$                                       <193>
2130 INPUT"NEUE[SPACE]STARTADRESSE[SPACE](MUSS
   [SPACE]IM[SPACE]ROM-BEREICH[SPACE2]LIEGEN
   :";A$                                           <134>
2140 GOSUB 1200:IF D=-1 OR D<AN THEN 2100          <235>
2150 I=D:GOTO 2040                                 <054>
2499 REM ROM STARTEN                               <106>
2500 GOSUB 1600                                    <033>
2510 POKE K+8,160:POKE K+29,0                      <235>
2515 POKE K+12,160                                 <142>
2520 SYS K                                         <202>
2525 K=828                                         <133>
2530 POKE K+8,96                                   <074>
2540 POKE K+29,128                                 <179>
2550 POKE K+12,INT(AN/256)                         <249>
2560 SYS K                                         <243>
2565 PRINT"[CLEAR]":PRINT">NEUES[SPACE]ROM
   [SPACE]AKTIVIERT":PRINT                         <200>
2570 POKE 1,53:INPUT"RESET[SPACE](J/N)";A$
   :IF A$="J"THEN SYS 64738                        <233>
2580 GOTO 500                                      <060>
2999 REM MONITOR FUER EINZELNE SPEICHERSTELLEN
                         <111>
3000 PRINT"[CLEAR]":PRINT"***[SPACE]MONITOR
   [SPACE]***":PRINT:PRINT                         <148>
3010 INPUT"STARTADRESSE:";A$:GOSUB 1200            <015>
3020 I=D-OF:IF I=-1 THEN 3010                      <093>
3030 D=I+OF:GOSUB 1300:PRINT A$;"[SPACE]";
   :D=PEEK(I):GOSUB 1300:PRINT A$;"[SPACE4]";
                         <229>
3040 INPUT":";A$                                   <142>
3050 IF A$=CHR$(13)THEN 3070                       <132>
3060 GOSUB 1200:IF D=-1 OR D>255 THEN 3090 <154>
3065 POKE I,D                                      <084>
3070 I=I+1:GOTO 3030                               <177>
3090 PRINT"ENDE[SPACE](J/N)"                       <175>
3095 GET A$:IF A$="N"THEN 3010                     <002>
3096 IF A$<>"J"THEN 3095                           <126>
3100 GOTO 500                                      <070>
3500 POKE 53247,0:END                              <107>
3599 REM WERTE FUER KERNAL RAM                     <002>
3600 AN=14*4096:EN=0:OF=8*4096:POKE 53246,0
   :RETURN                                         <035>
3699 REM WERTE FUER BASIC ROM                      <025>
3700 AN=10*4096:EN=12*4096:OF=4*4096
   :POKE 53246,1:RETURN                            <050>
3999 REM EIGENER MONITOR                           <100>
4000 PRINT"[CLEAR]":PRINT:PRINT"[SPACE7]EIGENER
   [SPACE]MONITOR"                                 <048>
4010 PRINT:PRINT:INPUT"STARTADRESSE[SPACE](HEX)
   :";A$                                           <205>
4020 IF LEN(A$)>4 THEN 4000                        <023>
4030 GOSUB 1200:IF D=-1 THEN 4000                  <031>
4040 SYS D                                         <185>
4050 GOTO 500                                      <255>
4999 REM ABSPEICHERN DES ROMS                      <106>
5000 GOSUB 1130:PRINT                              <192>
5010 PRINT TAB(7);"ABSPEICHERN[SPACE]DES[SPACE]
   NEUEN[SPACE]ROMS"                               <124>
5020 PRINT:PRINT:PRINT:INPUT"FILENAME:";A$ <140>
5030 OPEN 1,8,1,A$                                 <219>
5035 PRINT:PRINT"SAVING[SPACE]";A$                 <214>
5040 PRINT#1,CHR$(0);CHR$(96);                     <253>
5050 FOR I=6*4096 TO 32767                         <171>
5060 PRINT#1,CHR$(PEEK(I));:NEXT I                 <128>
5070 PRINT#1,CHR$(AN-INT(AN/256)*256);             <139>
5080 PRINT#1,CHR$(AN/256)                          <209>
5090 CLOSE 1:GOTO 500                              <030>
5999 REM GEAENDERTES ROM LADEN                     <142>
6000 GOSUB 1130:PRINT                              <172>
6010 PRINT TAB(10);"LADEN[SPACE]EINES[SPACE]
   NEUEN[SPACE]ROMS"                               <106>
6020 PRINT:PRINT:PRINT:INPUT"FILENAME:";A$ <120>
6030 OPEN 1,8,0,A$                                 <198>
6040 GOSUB 6100:GOSUB 6100                         <145>
6042 IF ST>0 THEN 6200                             <051>
6043 PRINT:PRINT"LOADING[SPACE]";A$                <000>
6050 FOR I=6*4096 TO 32767                         <151>
6060 GET#1,A$:A$=A$+CHR$(0)                        <241>
6070 POKE I,ASC(A$):NEXT I                         <090>
6080 GOSUB 6100:AN=ASC(A$):GOSUB 6100
   :AN=AN+256*ASC(A$)                              <041>
6090 OF=AN-6*4096:EN=AN+8192:CLOSE 1               <174>
6095 IF EN>65535 THEN EN=0                         <217>
6096 IF AN<14*4096 THEN POKE 53246,1:GOTO 500
                         <143>
6097 POKE 53246,0:GOTO 500                         <055>
6100 GET#1,A$:A$=A$+CHR$(0):RETURN                 <225>
6200 CLOSE 1:PRINT                                 <244>
6210 OPEN 1,8,15                                   <032>
6220 IF ST=64 THEN 6250                            <038>
6230 GET#1,A$:PRINT A$;:GOTO 6220                  <244>
6250 CLOSE 1                                       <083>
6260 GET A$:IF A$=""THEN 6260                      <039>
6270 GOTO 500                                      <180>
7990 REM AB HIER DATAS                             <252>
8000 DATA 64226,126,14014                          <209>
8001 DATA 232,134,198,201,133,144,4,201,141,144,
   3,76,66,235,157,119,2,72,152                    <150>
8002 DATA 72,160,0,196,2,208,13,185,41,251,221,
   119,2,240,11,200,192,176,208                    <087>
8003 DATA 243,104,168,104,76,66,235,200,185,41,
   251,201,133,144,4,201,141,144                   <140>
8004 DATA 238,236,137,2,176,233,157,119,2,232,
   134,198,76,15,251,133,76,207                    <080>
8005 DATA 34,36,34,44,56,13,137,76,79,65,68,134,
   76,73,83,84,13,138,83,65,86                     <160>
8006 DATA 69,135,82,85,78,13,139,83,89,83,136,
   63,70,82,69,40,48,41,13,140,83                  <202>
8007 DATA 89,83,54,52,55,51,56,13,133,136  <241>
8010 DATA 60223,3,552                              <016>
8011 DATA 76,226,250                               <227>
10000 DATA 0                                       <234>
```

Listing »ROM-Change« (Schluß)

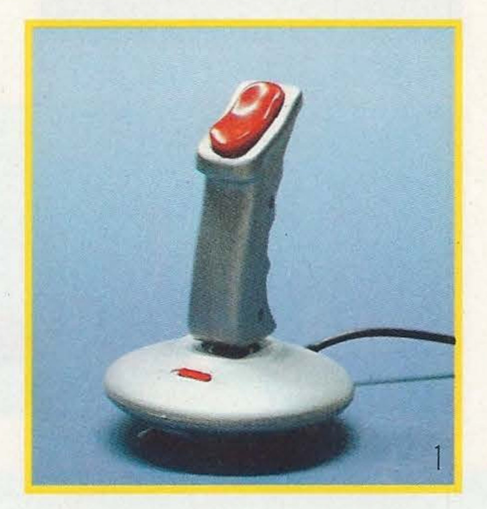


Name	Bildnummer	Saugnäpfe J/N	Dauerfeuer J/N	Links-Rechtshänder	Richtungen	Handhabung	Standsicherheit
Arkade	11	N	N	L/R	8	2	3
Arkade Professional	12 o.	N	N	L	4/8	3	1
Atari	8	N	N	R	8	2	3
Boss	15	N	N	L/R	8	2	3
Challenger	1	J	J	L/R	8	2	3
Competition pro	18	N	N	L/R	8	2	3
Competition pro	16	N	N	L/R	8	2	3
Commodore VC 1311	13	N	N	L/R	8	4	4
Gun Shot	5	J	N	L/R	8	2	2
Master Shot	14	J	J	L/R	8	1	1
Wico Three Way (1)	6	N	N	L/R	8	3	2
Wico Three Way (2)	9	N	N	L/R	8	3	2
Power Stick	12 u.	N	N	L/R	4	4	—
Quickshot 1	7	J	N	L/R	8	2	1
Quickshot 2	17	J	J	L/R	8	1	1
Tober	19	N	N	L/R	8	3	3
Atari Super Controller	3	N	N	L/R	8	2	2
Superstick	10	N	N	L/R	8	3	4
Erlkönig (Prototyp)	2	J	N	L/R	8	1	—
Sony	4	N	N	L/R	8	1	1

Das Angebot auf dem Joystick-Markt ist groß. Doch allzuoft vergeht bei der ersten Belastungsprobe die Lust am »Lustknüppel«. Wir sind der Frage nachgegangen, welcher Joystick nun der beste ist. Dabei wurden 20 Joysticks auf Herz und Nieren gerüft.

Gespannte Gesichter. Die Spielfigur läuft über den Bildschirm. Nur noch eine Gefahr ist zu überstehen. Endlich, endlich haben wir die Lösung vor Augen. Kurz nach links, dann rechts und ... knacks. Das typische Geräusch eines brechenden Joysticks ist zu hören.

Nur eine Szene aus unserem Joysticktest. Nicht immer spielt man mit einem Joystick, der genausoviel aushält, wie es in der Werbung versprochen wird und man es sich wünscht. Spiele wie »Soccer« und »Summer Games« haben schon manchem Joystick das »Leben« gekostet.


1


2


3

Stabilität	Leicht-Schwer-gängig	Feuerknopf Boden/Griff	Schalt-mechanik	Bemerkung	Test gesamt	Preis
2	mittel	1B	Mikroschalter	nur drei Standfüße	gut	69,–
2	leicht	2B	Mikroschalter	schlecht für Rechtshänder	empfehlens-wert	139,–
2	mittel	1B	Folienkontakt	—	befriedigend	39,–
2	leicht	1G	Metallstreifen	drehbarer Griff	gut	50,–
2	mittel	2G	Metallstreifen	—	gut	48,–
1	mittel	2B	Metallstreifen	sehr stabil	gut	69,–
1	mittel	2B	Mikroschalter	sehr stabil	empfehlens-wert	79,–
6	mittel	1B	Folienkontakt	zerbrechlich	mangelhaft	39,–
2	leicht	2G	Metallstreifen	—	gut	39,–
2	leicht	2G	Metallstreifen	besonders Standsicher	empfehlens-wert	49,–
1	leicht	1G 1B umschaltbar	Metallstreifen	austauschbarer Griff	gut	99,–
1	leicht	1G 1B umschaltbar	Metallstreifen	austauschbarer Griff	gut	99,–
4	mittel	2B	Gummifolie	zu klein für längeres Spielen	mangelhaft	59,–
2	leicht	1B 1G	Folienkontakt	—	gut	29,–
2	mittel	2G	Metallstreifen	Dauerfeuer kann unabsichtlich eingeschaltet werden	empfehlens-wert	39,–
2	schwer	1B 1G	Folienkontakt	Kabel kann im Fuß aufgewickelt werden	befriedigend	33,90
1	mittel	2B	Folienkontakt	Ist sehr gut in der Hand zu halten	gut	59,–
3	sehr leicht	1G	Metallstreifen	zu leichtgängig, daher nicht besonders Richtungsgenau	befriedigend	—
1	leicht	2G	auf Wunsch nicht erwähnt	—	empfehlens-wert	49,–
2	sehr leicht	2G	Mikroschalter	Resetschalter-Steuerung durch Handauflage	empfehlens-wert	69,–

Somit haben wir schon das erste und wohl auch wichtigste Kriterium in der Beurteilung eines Joysticks: die Stabilität und Verarbeitung.

Der zweite, nicht minder wichtige Beurteilungspunkt: die Handhabung und die Standfestigkeit. Zur Standfestigkeit sei nur erwähnt: Auch Saugnäpfe halten nicht immer, was sie versprechen. Das vierfache »Plopp«, wenn sich der Joystick von der Tischplatte löst, war auch mehr als einmal bei uns zu hören.

Bewährungsprobe

Nach dem Blick in das Innenleben wurden die Joysticks an unsere Tester gegeben. Getestet wurde mit den Spielen »Soccer«, »Summer Games« und »Zeppelin«. Die ersten beiden Spiele stellten sich als wahre »Joystick-Killer« heraus, das dritte aber war eine Herausforderung an die Richtungsgenauigkeit. Auf Richtungs- und Punktgenauigkeit kommt es auch in Zeichenprogrammen an. Nicht immer nützt Fingerspitzengefühl. So wurde auch die Verwendungsfähigkeit auf diesem Gebiet überprüft.

Beurteilt wurden außerdem die allgemeine Handhabung, die Standsicherheit und die Stabilität. Die Tester durften Noten von 1 bis 6 vergeben.

Uns standen bei dem Blick ins Innenleben einige Überaschungen bevor. Es ist verblüffend und erschreckend, wie primitiv und reparaturanfällig manche Joysticks aufgebaut sind. So wurde die Mechanik in die Wertung mit einbezogen. Mikroschalter erlauben eine hohe Richtungsgenauigkeit und sind sehr stabil. Deshalb wurden diese Schalter von uns mit 1 bewertet. Mit einer 2 wurden Metallstreifen-Schalter bewertet. Folien- und Gummischalter wurden wegen ihrer meist schlechten Qualität in der Genauigkeit und ihrer Reparaturanfälligkeit mit 4 und 5 beurteilt. Die Note ging normal in die Wertung ein, denn auch ein von der Technik mittelmäßiger Joystick kann seine Vorteile haben.

4



5 6 7 8 9



10 11 12 13 14





15 16 17 18 19

»Der« Joystick überhaupt!

Alle von uns haben ihren »Joystick-Traum«. Jeder von uns stellt an einen Joystick spezielle Anforderungen. Gut beweglich muß er sein, und leicht zu handhaben. Der Feuerknopf sollte einen Druckpunkt haben. Dauerfeuer und Reset-Knopf könnten auch dabei sein. Ob nun Saugnäpfe oder »Joystick ohne Boden«, gefallen soll er natürlich auch.

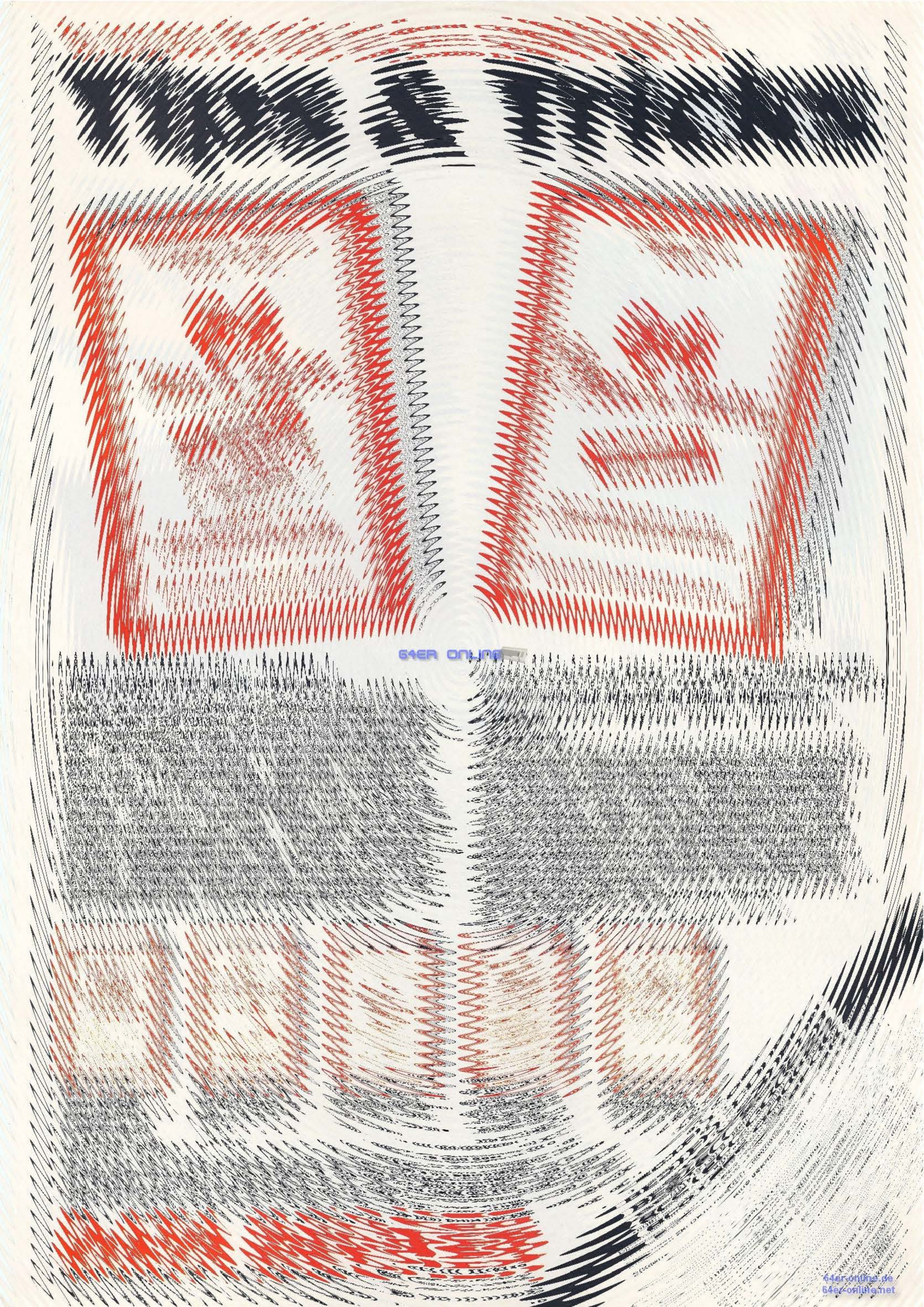
Wir halten sechs Joysticks für empfehlenswert, acht für gut, drei für befriedigend und zwei für mangelhaft.

Plant man die Anschaffung eines Joysticks, so sollte man auf jeden Fall selbst ausprobieren, »seinen« Joystick herauszufinden. Unsere Testtabelle soll nur eine Hilfestellung auf diesem fast unüberschaubaren Markt sein.

Kaum ein Händler wird erlauben, daß man einen Joystick stundenlang ausprobiert oder gar aufschraubt.

Die Gesamtnote des Tests ergibt sich aus den einzelnen Zensuren. Alle einzelnen Bewertungen sind gleich gewichtet. Zieht man die Noten zusammen, so hat man zwei eindeutige Sieger. Doch testen Sie nur selbst. Sind es auch Ihre Sieger?

(rg)





Graphics-Basic

Ein neuer Stern am Himmel der Befehlserweiterungen. Obwohl der Name sich auf die Grafik bezieht, gehen die zusätzlichen Befehle weit über diesen Bereich hinaus.

Bild 1. Hochauflösende Grafik — in reinem Basic programmiert

Bild 2. Die Diskette enthält viele Demo-Programme

Wie die Königstochter im Märchen, so bedarf es bei der Programmierung des Commodore 64 fast eines Prinzen, um seine schlummernden Fähigkeiten zu wecken. Unser Prinz heißt Graphic-Basic und kommt von HESware.

Haben Sie sich in Anbetracht eines perfekten Spielprogramms, schon gefragt: Wie machen die das? Die Antwort ist relativ einfach, denn durch konsequentes Ausnutzen aller verfügbaren Hilfsroutinen und vieler Programmiertricks gelingen solche Meisterstücke. Einige dieser Tricks und Hilfen sind in Graphic-Basic zusammengefaßt. Das Resultat sind Graphic-Basic-Programme, einfach programmiert, kurz und leistungsfähig. Das Beste aber ist, daß kaum jemand den Unterschied zu reinen Maschinenprogrammen feststellen kann. Von der Leistungsfähigkeit dieser zirka 90 Mark teuren Befehlserweiterung kann man sich sofort überzeugen, denn viele Beispielprogramme (Bild 1 und 2) werden auf der Diskette mitgeliefert. Damit die Lektüre der Anleitung (in Buchform, 116 Seiten) etwas abwechslungsreicher wird, findet der Anwender noch zusätzliche Beispiele zum Abtippen im Handbuch. Auf diese Weise bleibt das gerade gelernte nicht nur Theorie, sondern kann am Bildschirm mit eigenen Augen verfolgt werden.

Eines sei vorweggenommen, der Befehlssatz des »Graphic-Basic« läßt kaum einen Wunsch offen. Vergessen sie den POKE-Befehl, denn von der einfachen Farbänderung bis zur imposantesten Grafik- und Tonprogrammierung, stehen für alle Zwecke eigene Befehle zur Verfügung. Einfach und einprägsam, aber leistungsfähig, das sind die »Graphic-Basic«-Befehle. Das gilt auch für die Spriteprogrammierung. Denn hier haben sich die Programmierer selbst übertroffen. Bereits die Spriteerstellung geht mit dem eingebauten Sprite-Editor fast wie von selbst, der Eingabekomfort ist beispielhaft. So ist beispielsweise das Sprite, das gerade entworfen wird, in wirklicher Größe, in Y-Richtung und in X-Richtung vergrößert, abgebildet.

Ebenso besteht die Möglichkeit, die ganze 24x21 Punktmatrix um beliebig viele Spalten beziehungsweise Reihen zu verschieben. Wer schon früher mit ähnlichen Programmen gearbeitet hat, wird diesen Vorteil zu schätzen wissen. Selbstverständlich können sowohl Single- als auch Multicolorsprites definiert werden, die Vorder und Hintergrundabfrage, sowie die Kollisionserkennung sind eingebaut. Ein Sprite-Editor wie dieser wird noch lange Zeit seinesgleichen suchen.

Doch was kann man nun mit seinem eben erstellten Sprite-Kunstwerk alles anfangen? Sehr viel, denn dem Benutzer steht eine Vielzahl von neuen, sinnvollen Befehlen zur Verfügung. Diese Befehle zeichnen sich durch eine interessante Besonderheit aus, denn sie sind allesamt Interrupt-gesteuert. Das heißt, Sprites bewegen sich nach einmaligem Aufruf selbständig. Der Effekt ist, daß jedes Basic-Programm mit höherer Geschwindigkeit weiterläuft. Mancher Befehlserweiterung geht hier schon langsam die »Luft« aus, nicht so »Graphic-Basic«. Diese Supererweiterung sammelt weitere Pluspunkte, etwa durch die Befehle »Copy Hires

Fortsetzung auf Seite 150

Oxford-Pascal Für den Commodore

Dieser leistungsfähige Compiler unterstützt den vollen Pascal-Standard — und noch etwas mehr.

Schon der erste Blick in das Handbuch zeigt es: Oxford-Pascal ist im Wesentlichen das gute alte TCL-Pascal, das es auch schon für die CBM-Computer gibt.

Oxford-Pascal wird auf einer Diskette zusammen mit einem Handbuch in englischer Sprache geliefert (eine Kassettenversion soll demnächst folgen). Der Text des Handbuches sowie die abgedruckten Beispielprogramme sind überwiegend wörtlich vom TCL-Pascal übernommen. Neu am Oxford-Pascal sind Erweiterungen für Grafik, Ton und Farbe speziell für den C 64.

Nach einem recht langen Ladevorgang meldet sich das System schließlich mit einem »READY«. Oxford-Pascal ist im Gegensatz zum UCSD-Pascal nicht menügesteuert, was in meinen Augen insbesondere für den Computeranfänger ein wesentlicher Vorteil ist.

Auf Nachfrage teilt das System mit, daß etwa 13 KByte freier Speicherplatz zur Verfügung steht. Das ist beachtlich: TCL-Pascal auf dem CBM 3032 bietet nur etwas mehr als 7 KByte an. Der Compiler und der Executer sind resident, es kann also gleich losgehen mit den ersten Gehversuchen. Programm eintippen (mit Zeilennummern, damit man leichter editieren kann), auf »L« tippen und sich das Programm langsam vorlesen lassen. Dabei findet eine Syntaxprüfung und eine Compilierung in den P-Code statt. Mit »RUN« kann dann gestartet werden. Währenddessen findet kein Rückgriff auf die Floppy statt, dadurch ist das System relativ schnell. Sollte ein Fehler im Programm sein, wird eine Fehlermeldung mit Nummer und Klartext eingefügt. Die Eingabe von »P« statt »L« bringt Listing und Fehlermeldungen auf den Drucker. Editierkommandos wie CHANGE, FIND und DELETE vereinfachen Fehlersuche und Programmänderungen.

Leider vermißt man eine DOS-Unterstützung. Das Inhaltsverzeichnis ist nur mit »LOAD"$",8« zu erreichen, dabei geht ein eventuell im Speicher vorhandenes Programm, das nicht vorher mit PUT auf der Diskette abgespeichert wurde, verloren.

Was tut man jetzt, wenn 13 KByte nicht ausreichen? Oxford-Pascal bietet die Möglichkeit, den residenten Compiler aufzugeben. Damit stehen dann etwa 32 KByte nur für den Programmquelltext zur Verfügung. Das Programm muß nach Fertigstellung abgespeichert werden, dann wird mit »COMP« der Diskcompiler aufgerufen, der ein Object-File erstellt, das mit »EX« ausgeführt werden kann. Diese Prozedur ist typisch englisch: Teatime ist angesagt. Die Zeit ist ausreichend. Interessanterweise ist der P-Code von TCL-Pascal mit dem von Oxford-Pascal aufwärtskompatibel. Ich konnte mein Stundenplanprogramm, das vom CBM 3032 compiliert wurde, sofort auf dem C 64 laufen lassen.

Die Abwärtskompatibilität ist natürlich nicht gewährleistet. Mit einigen Spezialitäten wurde den Möglichkeiten des Commodore 64 Rechnung getragen. Mit BORDER wird die Rahmenfarbe definiert, mit SCREEN die Bildschirmfarbe, mit PEN die Schreibfarbe. PAPER und INC setzen die Farben für die hochauflösende Grafik (der Multicolormodus wird nicht unterstützt), HIRES (1) schaltet die Grafik ein, HIRES (0) führt in den Textmodus zurück. WINDOW teilt den Bildschirm in einen Grafik- und einen Textteil ein. EXAMINE (X,Y) dient zum Testen, ob der Punkt (x,y) gesetzt ist. Für alles weitere ist der PLOT-Befehl zuständig. PLOT (F,X1,Y1,X2,Y2) hat in Abhängigkeit von F folgende Aufgaben:

Ist F = 0, so wird der Hintergrund auf die PAPER-Farbe gesetzt,
ist F = 1, so wird die Grafik gelöscht,
ist F = 2, so wird eine Linie von (X1,Y1) nach (X2,Y2) gezeichnet,
ist F = 3, so wird diese Linie gelöscht,
ist F = 4, so wird das Gebiet um (X1,Y1) gefüllt,
ist F = 5, so wird dieses Gebiet gelöscht.

Für das vollständige Löschen der Grafik ist also die Anweisungssequenz PLOT (0,0,0,0,0); PLOT (1,0,0,0,0) notwendig. Man beachte, daß von den neun Nullen acht lediglich aus formalen Gründen angegeben werden müssen. Der Pascalästhetiker wird dabei ein ungutes Gefühl haben. Durch den gewiß lobenswerten Versuch, die Anzahl der Spezialitäten gering zu halten, leidet die Selbstdokumentation doch erheblich.

Immerhin braucht sich der Programmierer nicht so viele neue Vokabeln zu merken. Es ist möglich, im PLOT-Befehl x-Werte zwischen 0 und 255 sowie y-Werte zwischen 0 und 199 anzugeben. Da der Bildschirm aber 320 Pixel breit ist, bleibt rechts ein Streifen, auf den man nicht zugreifen kann.

Falls eine Darstellung von Text und Grafik gleichzeitig gewünscht wird (ein Einblenden von Buchstaben in die Grafik ist nicht vorgesehen), so wird man mit dem WINDOW-Kommando, so nützlich es auch ist, nicht so ganz glücklich. Da die Umsteuerung des Bildschirms durch den Interrupt erfolgt, flackert er bei Diskettenzugriffen erheblich.

So mager und ungeschickt die Grafik auch ausgestattet ist, so vollständig ist der von Wirth geforderte Sprachumfang. Man wird nichts vermissen. Hier zeigt sich die Abstammung vom TCL-Pascal mit allen Vorteilen. Erwähnenswert ist auch, daß einige der von mir im Artikel »Für Schulen gerade richtig« (Computerjournal Januar/Februar 1983) monierten Fehler beseitigt sind. File-Identifier können jetzt als VAR-Parameter an Prozeduren und Funktionen übergeben werden (übrigens: im Apple-UCSD-Pascal geht das nicht einmal mit Funktionsnamen und trotzdem gibt es Schulen, die es im Unterricht verwenden), und Mengen, die Bestandteile eines Records sind, werden jetzt unter der WITH-Anweisung korrekt bearbeitet. Vermutlich hat es schon TCL-Versionen gegeben, die diese Fehler nicht mehr hatten, und dies wurde einfach übernommen.

Zusammenfassend muß festgestellt werden, daß man hier für 199 Mark ein vollwertiges Pascal-System in der Hand hat, das sowohl dem Lernenden als auch dem Anwender sehr viel zu bieten hat.

(Norbert Stüven/ev)

Info: Vertrieb in Deutschland durch Computer Plus Soft GmbH, Bahnstr. 22-26, 4220 Dinslaken

Die Turbo-Pascal-Story

Das Super-Pascal aus der Welt der CP/M- und MS-DOS-Personal Computer ist jetzt auch für den C 64 erhältlich — das CP/M-Modul macht's möglich.

Um Turbo Pascal auf dem C 64 nutzen zu können, sind Floppy-Laufwerk und CP/M-Modul notwendige Voraussetzungen. Zum optimalen Betrieb sollten Grundkenntnisse von CP/M-80, der Sprache Pascal und etwas Z80-Maschinensprache hinzukommen. Benutzt man einen Drucker mit Centronics-Schnittstelle, sollte man das Betriebssystem des C 64 und 6502 Maschinensprache kennen, denn mit den Interfaces, die Treibersoftware im Bereich C000 — $CFFF_{16}$ bereithalten, könnte man Überraschungen erleben.

Die Lieferung

Die Sendung von Heimsoeth Software, München, zum Preis von 225,72 Mark, enthält die Diskette und ein Handbuch in englischer Sprache von etwa 300 Seiten Umfang. Das Handbuch versteht sich nicht als Einführung in Pascal, sondern beschreibt in drei Teilen den Gebrauch des Turbo Systems, die Besonderheiten für einzelne Betriebssysteme wie CP/M-80 und 86 sowie MS/PC-DOS, die Sprachelemente (Syntax) von Turbo Pascal. Den Schluß bilden recht brauchbare Übersichten zum Pascal, zum Compiler und den Fehlermeldungen. Die Diskette enthält die Files

— **TURBO.COM,** also den eigentlichen Pascal-Compiler, der laut STAT-Meldung etwa 31 KByte belegt,

— **TURBO.OVR,** eine Routine, die Programmoverlaying im CP/M ermöglicht. Damit können auch Programme, die mehr als den zur Verfügung stehenden Speicherplatz (zwischen 6100 und 11600 Byte je nach Konfiguration) benötigen, gefahren werden.

— **TURBO.MSG** ist ein Textfile, das die 100 Fehlermeldungen im englischen Wortlaut enthält. TURBO.MSG kann beim Initialisieren wahlweise hinzugeladen werden und belegt dann etwa 1400 Byte. Wer will, kann sich alle Fehlermeldungen ins Deutsche umschreiben. Ohne TURBO.MSG werden nur die Nummern der Fehler nach dem Compilieren angegeben. Ihre Bedeutung schaut man dann im Handbuch nach. Leider habe ich noch keine Möglichkeit gefunden, TURBO.MSG zwischenzeitlich nachzuladen. Man muß TURBO.COM erst mit Q verlassen und dann mit »TURBO« neu laden.

— **TLIST.COM** ist eine Druckerroutine, die es nicht nur ermöglicht, Pascal-Quellcode mit vorgestellten Zeilennummern auszudrucken, sondern auch Pascal-Schlüsselwörter durch Unterstreichen zu markieren.

— **READ.ME** gibt eine Erklärung darüber, warum Run-Time-Fehler in Overlay-Programmen nur schwer zu lokalisieren sind, falls die Overlay-Area mehr als ein Programm enthält. Die beiden Programme TINST.COM und CALC.PAS, die man auf Lieferungen für andere CP/M-Versionen (zum Beispiel Osborne und TA) findet, waren nicht vorhanden. Ich erkläre mir dies damit, daß das Demoprogramm zur Tabellenkalkulation nicht in den Speicher des C 64 paßt und das Tastatur-Installationsprogramm sich für die auf den C 64 zugeschnittene Version von selbst erübrigt.

Starten des Systems

Wer liest sich, wenn er Software erhält, schon ein ganzes Handbuch durch, um danach erst in die Praxis einzusteigen? Also schob ich nach einem Blick auf die ersten paar Seiten des Handbuches mein CP/M-Modul ein, legte die CP/M-Arbeitsdiskette ins Laufwerk und zählte nach »LOAD "CPM", 8« und »RUN« die 28 Sternchen, bis sich CP/M meldet. Dies dauert in der Regel etwa eine Minute, kann aber auch mal ins Auge gehen, je nachdem wie sorgfältig das Modul eingesteckt wird. Nach Meldung des CP/M stieg die Spannung, denn ich legte die Turbo-Diskette ein und tippte, da Turbo Pascal als CP/M-Kommando gestartet wird, einfach TURBO ein. Mit Freude stellte ich fest, daß sich das Laufwerk in Bewegung setzte. Meine Freude wurde etwas getrübt, dauerte der Ladevorgang doch fast 2 Minuten (genauer 106 Sekunden). Damit verbraucht das Hochfahren des Systems also insgesamt fast 3 Minuten. Dies liegt jedoch nicht am CP/M, sondern am langsamen Arbeiten des 1541-Laufwerks. Immerhin wurde mein Warten mit einer recht klaren Turbo-Meldung begrüßt, wobei die gelbe Schrift auf schwarzem Hintergrund sich wohltuend gegenüber dem Commodore-Blau abhob.

Nun mußte wieder das Handbuch zu Rate gezogen werden, denn mit einer simplen Meldung wollte ich mich nicht begnügen. Zumindest ein kleines Programm sollte laufen. Also wurde nach Eingabe von »N« (keine Fehlermeldung im Wortlaut) mit »W« eine Arbeitsdatei (Workfile) angelegt. Dabei muß man zunächst den Namen angeben. Will man eine bereits existierende Datei bearbeiten, so wird diese jetzt geladen. »DIR« zeigt das Inhaltsverzeichnis der Diskette. Da noch kein .PAS-File existierte, erklärte die Meldung »NEW FILE«, daß nun eine neue Arbeitsdatei eingerichtet wird. Nach Eingabe von »E« meldete sich ein auf den ersten Blick recht komfortabler Editor, der eine Teilmenge der Wordstar-Befehle bereithält und 80 Zeichen pro Zeile unterstützt. Leider wurde die Repeat-Funktion für die Corsorbewegungen vergessen. Was mich zuerst sehr erschreckte, als ich daran ging zwei Tippfehler meines ersten eingetippten Programms auszumerzen, war die Tatsache, daß der Cursor alle Zeichen löschte, die er überstrich. War damit meine ganze Eingabe umsonst? Konnte man mit Turbo auf dem C 64 gar nicht arbeiten? Verwundert hätte mich das nicht, denn bisher findet man ja nur ganz wenig CP/M-Software, die an das eigenwillige CP/M des C 64 zufriedenstellend angepaßt ist. Im ersten Moment war ich ratlos und las wieder Seite um Seite im Handbuch. Doch dies ist allgemein gehalten und kann, will es nicht zu einer Enzyklopädie ausarten, nicht auf alle CP/M-fähigen Computer eingehen.

Also raus aus dem Editor mit CTRL K, CTRL D und mit E wieder hinein, um mein Glück aufs Neue zu probieren. Aber, welche Freude! Da stand mein Programm TEST1, so wie ich es eingegeben hatte.

Nun war die Sache wohl klar: Beim Schreiben des Editors hat man vermutlich vergessen, die Routine BASIN des C 64-Betriebssystems sorgfältig umzuschreiben. Deshalb

wird an der Stelle, wo die Farbinformation des Zeichens unter dem Cursor ins Farb-RAM geschrieben wird (also im Bereich $E45F-$EB7F), genau dies einfach vergessen.

Da ich eine sehr »frische« Disk der Version 2.0 A vorliegen hatte, dürfte dieser vom Prinzip her kleine, in der Wirkung aber störende Fehler, in nächster Zeit behoben sein. Das dem so ist, sicherte auch die Münchener Vertriebsfirma telefonisch zu. Eine Weile wird man dennoch warten müssen, denn Kalifornien ist immer noch weit und bis die Eingaben aus der »Provinz BRD« berücksichtigt werden, wird einige Zeit ins Land gegangen sein.

Turbo Pascal lehnt sich eng an den durch Jensen und Wirth festgelegten Pascal-Standard an. Folgende Unterschiede, die wohl nur den eingefleischten »Wirth Pascaler« tiefer berühren werden, sind zu vermerken:

— Der Programmkopf ist optional gestaltbar.

— Im Deklarationsteil muß die Reihenfolge und Anzahl der Vereinbarungen nicht sklavisch eingehalten werden.

— Als Label können Zahlen und Namen dienen.

— Die verpönte GOTO-Anweisung darf nur im laufenden Block verwendet werden.

— Das »Packen« insgesamt fehlt, da der Turbo-Compiler ohnehin einen sehr dichten Code generiert. Während PACKED nichts bewirkt, rufen die Prozeduren PACK und UNPACK eine Fehlermeldung hervor.

— Wie in vielen anderen Versionen sind auch Funktionen und Prozeduren als Parameter nicht zugelassen.

Hält nun der Name Turbo das, was er verspricht? Diese Frage kann man in zweierlei Hinsicht guten Gewissens bejahen: Einerseits ist die Compilierzeit auf Grund des RAM-residenten Compilers erheblich schneller als bei anderen, vergleichbaren Pascal-Systemen. So ist der Turbo-Compiler bis zu zehnmal schneller als der 2-Pass-Compiler des Apple-UCSD-Systems. Aber auch die Abarbeitung von Programmen ist auf Grund der Tatsache, daß der Turbo-Compiler nicht nur 8080-Code generiert, sondern auch die vielen zusätzlichen, leistungsfähigen Befehle des Z80 optimal nutzt, meist schneller als bei vergleichbaren Systemen. Daß dabei natürlich der mit nur 2 MHz betriebene Z80 des C 64 langsamer ist als der mit 4 MHz betriebene Z80 anderer CP/M-Computer, liegt auf der Hand.

Wenn man nun in Pascal arbeitet und das 1541-Laufwerk mal nicht benutzt, kann man manchmal vergessen, daß man eigentlich vor einem Computer der unteren Preisklasse sitzt. Dieses Gefühl, vor einem wesentlich teureren Gerät zu sitzen, wird durch die elfstellige Rechengenauigkeit nur noch bestärkt. So ergibt 1.0000001 27mal mit sich selbst quadriert, immerhin noch den Wert 674423,34803. Dies entspricht einem Fehler von 0,016 Prozent im Vergleich zum 15stelligen Wert von 674530,470741078 einer CDC-Cyber-Anlage (Die Fehlerquote bei Basic liegt bei über 100 Prozent!). Welches andere Pascal rechnet dazu die ersten 1 000 Primzahlen in knapp 3 Sekunden aus?

	BENCH1	BENCH2	BENCH3	BENCH4	
APPLE/UCSD	0,8	0,4	9,9	11,1	(s) 2 MHZ 6502
C64/TURBO	0,8	0,4	10,4	12,6	(s) 2 MHZ Z80
TA PC/TURBO	0,7	0,4	5,3	6,3	(s) 4 MHZ Z80

C 64 — Turbo Pascal im Zeitvergleich

Neben dem Vorzug effizienter Compilierung und hoher Ausführgeschwindigkeit, bietet Turbo Pascal selbstverständlich auch nützliche Erweiterungen an, die über das Standard-Pascal hinausgehen. Dabei erkennt man schon vom Handbuch her die Vorzugsstellung von IBM; denn für den IBM und Kompatible finden sich zusätzlich Grafik- und Soundbefehle als »IBM-Goodies«. Vielleicht wird man in Turbo auch mal »C 64 Goodies« finden, falls der Hersteller Borland International den C 64-Markt und die C 64-Benutzer Borland entdecken.

Um den Rahmen nicht ganz zu sprengen, seien hier nur einige Erweiterungen knapp skizziert:

— Strings sind voll implementiert. Dazu kommen acht Funktionen/Prozeduren, die den Umgang mit Strings erleichtern.

— Eine Gruppe von Funktionen unterstützt den direkten Zugriff auf absolute Speicheradressen ähnlich PEEK, POKE und VARPTR.

— Durch das Schlüsselwort EXTERNAL werden externe Unterprogramme (speziell in Maschinensprache) in das laufende Programm eingebunden. Analog ist es möglich, durch das Schlüsselwort INLINE Maschinenprogramme im Z80-Code direkt als Prozeduren in Pascal-Programme zu schreiben.

— Dazu bietet der Compiler zehn Direktanweisungen. Die Compileroption C beispielsweise wandelt Pascal-Programme in CP/M 80-Befehle um. Diese neuen CP/M-Kommandos in Verbindung mit dem SUBMIT, besser noch mit XSUBMIT des CP/M, öffnen die Türe weit, um auch komplexe Probleme elegant und übersichtlich zu lösen.

Abgesehen von einzelnen behebbaren Mängeln ist Turbo Pascal die ideale Erweiterung und Ergänzung zum CP/M-Modul des C 64. Durch das langsame Diskettenlaufwerk ist jedoch am kommerziellen Einsatz primär nicht zu denken.

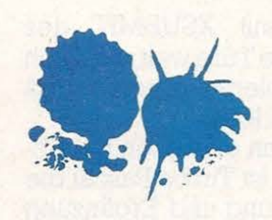
Wo das C 64/Turbo-Pascal-System jedoch optimal hingehören könnte, ist der gesamte Komplex der Aus-, Fort- und Weiterbildung, sei es im privaten Bereich, wo man ein preiswertes und schnelles Pascal-System sucht, oder im öffentlichen Bildungsbereich.

Mit dem schnellen Compiler entstehen keine Geduldsproben durch einen stundenlagen Kampf gegen Tippfehler. Syntax- wie Laufzeitfehler werden rasch entdeckt, denn hat der Compiler einen Fehler gemeldet, so wird nach dem Drücken der CLR/HOME-Taste das Programm im Editor gelistet, und der Cursor steht an der Fehlerstelle.

Wenn jetzt die Stunde genutzt wird, können der C 64 und Turbo-Pascal eine für alle Beteiligten gewinnbringende Ehe eingehen. Was im Moment noch fehlt, ist eine deutsche Einführung in das Turbo-System, unter Berücksichtigung der Eigenheiten des C 64. Sonst stehen wirklich alle Möglichkeiten offen. Denn wo ist ein anderes System, das zu einem Preis von etwa 2500 Mark nicht nur Drucker, Laufwerk und Zentraleinheit bereithält und eine Einführung in das weitverbreitete Betriebssystem CP/M-80 gestattet, sondern auch darüber hinaus eine der modernsten Sprachen, nämlich Pascal, mit allen Möglichkeiten der modularen Top-Down-Programmierung (weshalb Pascal in einzelnen Bundesländern als verpflichtend für den Informatikunterricht der S II gilt) ohne Abstriche und Einschränkungen zur Verfügung stellt? Hier ist eine große Chance, entweder für Commodore oder auch für andere Hersteller, das leider zu früh »gestorbene« CP/M-Modul für den C 64 wieder zum Leben zu erwecken.

(Hans-Ulrich Korzilius/ev)

Vertrieb für Deutschland, Heimsoeth-Software, Frauenhoferstr. 13, 8000 München 5, 089-264060



Was bringt die Lern-Software?



Das Leben ist voller Entscheidungen. So muß mancher Schüler darüber nachdenken, ob er den Nachmittag mit Lernen oder mit seinem Computer verbringt. Warum nicht beides miteinander verbinden?

Ist erst einmal ein Computer im Haus, dann bleibt das Lernen meist auf der Strecke. Fallen die Schulnoten in den Keller, muß ein Nachhilfelehrer her. Dieser kostet dann zwischen zwanzig und dreißig Mark pro Stunde, wenn nicht sogar mehr. Doch was soll man machen?

Ob Lernsoftware einen Nachhilfelehrer ersetzen kann, ist wohl von Fall zu Fall unterschiedlich. Die besseren Programme dürften aber zumindest in diese Richtung gehen.

Schenkt man der Werbung oder der Verpackung Glauben, so ist jedes Programm das »non plus ultra«. Wirft man dann einen Blick hinter die Kulissen, kann man von der einfachsten Programmierung bis hin zu gut durchdachten Programmen alles entdecken.

Besorgte Eltern, die ihren Kindern Lernprogramme kaufen wollen, stehen oft auf verlorenem Posten. Meist fehlt die Sachkenntnis zu dem Lehrstoff, zum anderen sehen viele Programme auf dem Bildschirm interessanter aus, als sie eigentlich sind. Und wer wollte schon den pädagogischen Nutzen anhand einer kurzen Vorführung beurteilen?

Deshalb beginnt das 64'er Magazin ab dieser Ausgabe mit einer Serie, die sich mit Lernsoftware beschäftigt. Sie soll zeigen, welche Programme qualitativ das halten, was sie versprechen, und an welche Anwendergruppen sie sich richten.

(rg)



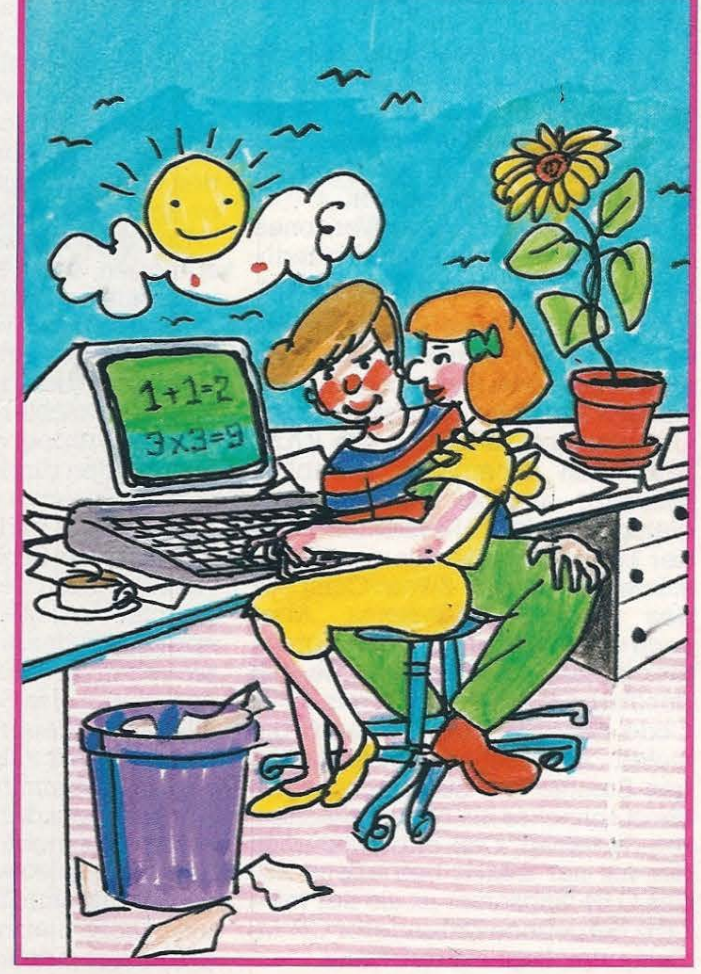




Bild 2. Komponieren mit dem Melodienschreiber

Melodienschreiber und Musik-Synthesizer

Melodienschreiber und Musik-Synthesizer sollen Lernprogramme für den angehenden Musiker sein. Lohnt es sich, Gitarren oder Geigenunterricht dafür aufzugeben?

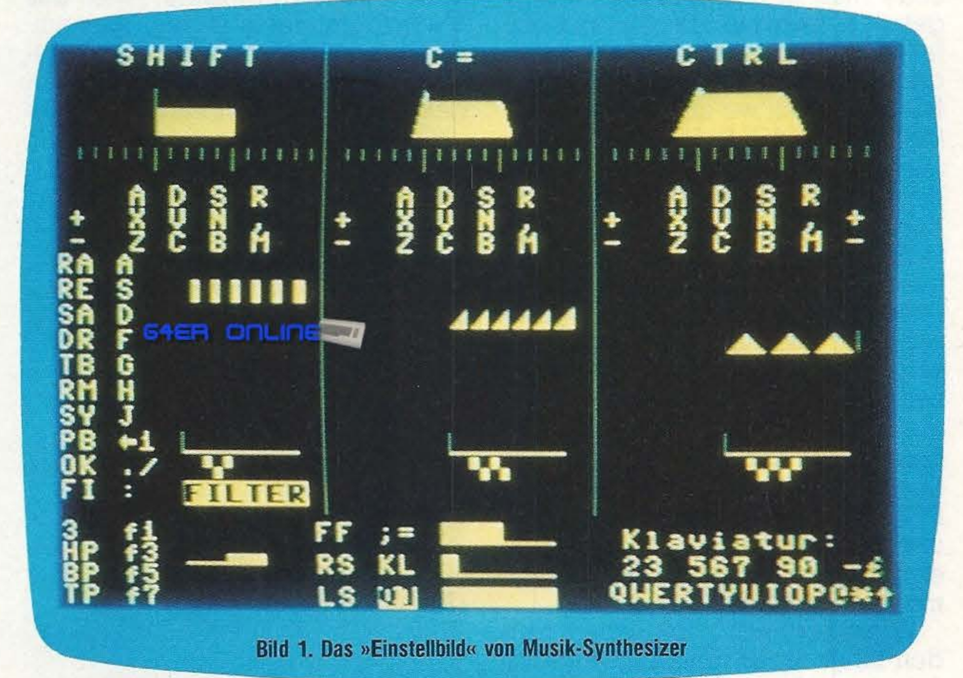


Bild 1. Das »Einstellbild« von Musik-Synthesizer

Das Programm »Musik-Synthesizer« gibt es in je einer Version für das Commodore Basic V.2.0 und Simons Basic. Ist das Programm geladen, erscheint bei der Simons Basic-Version nach Nennung des Autors das einzige Bild des Programms (Bild 1).

Trotz einiger Ungenauigkeiten des Handbuches lassen sich bei intensiver Lektüre interessante Tonzusammenstellungen erzeugen. Ein lästiger Nachteil: Zur Einstellung und Erzeugung von Tönen muß man immer eine der drei Kontrolltasten bedienen.

Das Programm erlaubt, durch Drücken einer Taste (plus Kontrolltaste) die Hüllkurven der drei verfügbaren Stimmen zu verändern. Dabei kann man Attack, Decay, Sustain und Relase ändern. Dies wird auch grafisch angezeigt. Weiterhin hat man die Auswahl zwischen Rechteck-, Sägezahn- und Dreieckswellenform sowie Rauschen. Man kann eine Ringmodulation und Synchronisation zuschalten, um andere Stimmen zu beeinflussen.

Zur Änderung der Oktaven stehen zwei Tasten zur Verfügung, wobei die oberste Oktave nicht über die ganze Tastatur reicht.

Eine weitere Möglichkeit bietet ein zuschaltbarer Filter, den man als Hochpaß, Bandpaß und/oder Tiefpaßfilter schalten kann. Das mehrstimmige Spielen von Tönen beschränkt sich auf das Spielen eines Tons mit maximal drei Schwingungsformen. Die Version in Basic V.2.0 ist bis auf die Grafik mit der Simons Basic-Version identisch.

Das Programm unterstützt das Musikverständnis und erfüllt damit die gestellte Aufgabe, aber es gibt mittlerweile weitaus bessere Synthesizer und Musikprogramme.

Das Programm Melodienschreiber läuft nur mit Simons Basic. Für jemanden mit wenig Erfahrung in der Musiklehre ist das Programm fast ungeeignet. Das Handbuch setzt ein recht großes Grundwissen voraus.

Es beginnt mit dem Thema, das eigentlich an den Schluß gehört: Mit dem Ändern der Melodie. Es sollte deshalb gleich zu Punkt 3 übergegangen werden. Dort werden die Eingabemöglichkeiten einer Melodie erklärt. Dabei stehen drei Arten zur Auswahl. Bei der manuellen Eingabe zeigt das Menü die Töne und die Tonlänge an. Die Eingaben erfolgen alle durch Input-Befehle. Die zweite Variante benutzt die beiden oberen Tastenreihen als Klaviertastatur. Im Hintergrund ist ein Metronom zu hören. Wählt man diese Eingabeart, sollte man ein wenig Klavier spielen können.

Die letzte Funktion erlaubt, das Tempo selbst zu bestimmen. Die Eingabe wird wie gewohnt gehandhabt. Nach jeder Eingabe erscheint ein Notenbild, bei dem man eventuelle Fehlinterpretationen des Computers erkennen kann.

Auch dieses Programm hat seine Grenzen. Bei mehr als 200 Einzeltönen wird die Zeitspanne zum Auffinden der Daten zu groß. Um trotzdem mehr Töne einzugeben muß die Basic-Zeile 7 geändert werden.

Dieses Programm ist für den Hobby-Musiker sicher eine feine Sache, doch als Lernprogramm erscheint es ungeeignet. Besonders die mangelhaften »Handbücher« fallen dabei schwer ins Gewicht.

(Sven Seynsche/rg)

So macht man Basic-Programme schneller



Diesmal wollen wir den Basic-Programmen, was die Geschwindigkeit anbelangt, mit einem Ausflug in die Assemblerprogrammierung, auf die Sprünge helfen.

Hier soll Ihnen kein Maschinensprache-Kurs zugemutet werden. Doch ein Programm in Maschinensprache besteht genauso aus Befehlen, Adressen und Variablen, wie ein Basic-Programm, nur sind sie in einem speziellen Zahlencode geschrieben. Dieser Zahlencode muß in den Arbeitsspeicher geladen werden. Die für uns einfachste Möglichkeit besteht darin, die Zahlen in den Speicher hineinzuPOKEn. Damit wir aber nicht unmäßig viele POKE-Befehle schreiben müssen, legen wir alle Code-Zahlen hinter DATA-Befehle und holen sie dann mit READ in eine einzige POKE-Schleife. Ich sage das deswegen, weil dieses Einlesen natürlich nicht zu dem Testprogramm gehören darf, dessen Laufzeit wir messen wollen. Das Testprogramm selbst sitzt zwischen den drei Zeilen der »Stoppuhr«. Das heißt, genauer gesagt sitzt das Programm in den Speicherzellen, in die wir es hineinPOKEn. Aber zwischen der Stoppuhr rufen wir es auf, der dem RUN entsprechende Befehl bei Maschinensprache heißt SYS.

Wie Sie gleich noch sehen werden fängt unser Testprogramm ab Speicherzelle 7168 an. Das Ganze sieht dann so aus:

```
10 TI$="000000"
20 PRINT CHR$ (147)
30 SYS 7168
1000 POKE 214,18:PRINT:PRINT
TI/60 "SEKUNDEN":END
```

Ab Zeile 2000 setzen wir jetzt das Programm, welches uns das Maschinenprogramm einliest. Um mit dem Einlesen zu beginnen, setzen wir noch eine Umleitung vor das Meßprogramm:

```
5 GOTO 2000
```

In Zeile 2000 löschen wir den Bildschirm. Zeile 2010 und 2020 und 2030 liest die Codezahlen, die von Zeile 2050 bis 2090 stehen, und POKEt sie in die Speicherplätze 7168 bis 7200.

Die Codezahlen sind für beide Computer fast identisch, nur die Adressen sind verschieden. Deshalb sind Zeile 2060 und 2080 beim C 64 anders als beim VC 20.

Sobald die Zahlen eingelesen sind, können Sie das Meßprogramm mit dem Befehl GOTO 10 (direkt eingetippt) starten.

Im Abdruck unten wird das etwas eleganter gemacht. Zuerst meldet das Programm das Ende des Einlesens (Zeile 2100 und 2101). Dann kommt die Anweisung, wie das Meßprogramm zu starten ist, nämlich durch Drücken irgendeiner Taste, die durch eine GET-Schleife abgefragt wird. Wenn eine Taste gedrückt wird, springt das Programm auf Zeile 10 (das geschieht in Zeile 2160).

```
5 GOTO 2000
10 TI$="000000"
20 PRINT CHR$(147)
30 SYS 7168
1000 POKE 214,18:PRINT:PRINT
TI/60 "SEKUNDEN":END
2000 PRINT CHR$(147)
2010 FOR A=7168 TO 7200
2020 READ B
2030 POKE A,B
2040 NEXT
2050 DATA 162,0,169,1,157
2060 DATA 0,30,169,6,157,0,150
(2060 DATA 0,4,169,14,157,0,216)
2070 DATA 232,224,0,208,241,169,1,
157
2080 DATA 254,30,169,6,157,254,150
(2080 DATA 254,4,169,14,157,254,216)
2090 DATA 232,224,120,208,241,96
2100 PRINT"DAS MASCHINEN-
PROGRAMM"
2110 PRINT"IST JETZT EINGELE-
SEN.":PRINT
2120 PRINT"ZUM STARTEN DES
PRO—"
2130 PRINT"GRAMMS "CHR$(18)
"TASTE"CHR$(146)"DRUECKEN"
2140 GET A$
2150 IF A$=" "THEN 2140
2160 GOTO 10
```

So, inzwischen haben Sie sicher Ihre Überraschung gehabt! 0,066 Sekunden Laufzeit beim C 64 und 0,033 beim VC 20.

Ich hoffe, daß ich Sie mit dem Virus der Maschinensprache infiziert habe.

Wir wollen im Folgenden ein paar arithmetische Funktionen untersuchen und beschleunigen. Als erste nehmen wir uns in **Version 17** die Multiplikation vor. Die Messung der Laufzeit erfolgt auf dieselbe Weise wie bei allen Programmen vorher auch. Deshalb bleiben die Zeilen 10, 20 und 1000 gleich. Die Multiplikation selbst soll 300 mal ausgeführt werden (Zeile 30). Dann wird das Ergebnis gedruckt (Zeile 60).

```
30 FOR Z=1 TO 300
50 NEXT
60 PRINT A
```

Als Multiplikation nehmen wir den Extremfall einer kurzen Zahl multipliziert mit einer langen.

```
40 A=3*0,123456789
```

Nach RUN bleibt der Bildschirm zuerst leer, bis dann nach 11,85 (14,15) Sekunden das Ergebnis der Multiplikation und die Laufzeit ausgedruckt wird.

In **Version 18** vertauschen wir die beiden Zahlen, die in Zeile 40 multipliziert werden.

```
40 A=0.123456789*3
```

Diese einfache Manipulation bringt natürlich nach Adam Riese dasselbe Resultat wie vorher, aber die Laufzeit ist kürzer. Wir gewinnen beim VC 20 0,37 Sekunden, beim C 64 0,44 Sekunden. Dieser Gewinn ist nicht überwältigend, aber überraschend. Aber denken Sie nach!

Wie ist das, wenn Sie so eine Multiplikation auf dem Papier durchführen? Da ist die Rechnung im zweiten Fall auch einfacher. Der Computer hat genau dasselbe Problem.

In **Version 19** nützen wir noch eine kleine Eigenheit der Commodore-Computer aus, die auf ihre amerikanische Herkunft zurückzuführen ist. Bei den Angelsachsen ist es nämlich erlaubt, eine Null vor dem Dezimalpunkt wegzulassen. Beim Computer dürfen wir das auch. Obwohl das mit der Multiplikatikon direkt nichts zu tun hat, bietet sie uns doch eine gute Gelegenheit, die Einsparung durch das Weglassen der Null auch zeitlich zu messen. Also, Zeile 40 sieht jetzt so aus:

```
40 A=.123456789*3
```

Das bringt nicht sehr viel, 0,17 Sekunden beim VC 20, 0,20 Sekunden beim C 64. Aber Kleinvieh macht auch Mist.

Eine ähnliche Verbesserung, die wir hier nicht ausprobieren, wird erzielt durch den Ersatz einer alleinstehenden Null durch einen Punkt, zum Beispiel:

statt IF X=0 THEN —
jetzt IF x=. THEN —

Eine gewaltige Beschleunigung erfährt das Multiplikationsbeispiel, wenn wir die Regel 1 anwenden und die Variablen vordefinieren.

Regel 7
★Bei einer Multiplikation soll die längere Zahl vor der kürzeren stehen (langer Multiplikant, kurzer Multiplikator).
★Eine einzelne Null wird durch einen Punkt ersetzt, eine Null vor dem Dezimalpunkt wird weggelassen.

In **Version 20** ersetzen wir in Zeile 40 beide Zahlen durch Buchstaben, die wir in einer neuen Zeile 25 diese Werte zuweisen.
25 B=.123456789:C=3
40 A=B*C
Dieser Lauf bleibt nach 1,23 (1,48) Sekunden stehen, das heißt wir gewinnen 10,25 (12,23) Sekunden (von 11,48!). Also bitte Regel 1 unbedingt beachten!

Eine andere betrachtenswerte arithmetische Funktion ist das »Potenzieren« (Quadrat-/Kubikzahlen), ausgelöst durch das Zeichen ↑. **Version 21** erzielen wir durch Löschen der Zeile 25 und Abänderung der Zeile 40:
40 A= 4 ↑ 3
»Vier hoch drei« ergibt 64 und braucht 8,81 (10,53) Sekunden.

In **Version 22** wollen wir sehen, ob vordefinierte Variable auch so einschlagen, wie bei der Multiplikation.
25 B=4:C=3
40 A=B↑C
Man kann sich doch auf nichts verlassen! Diesmal sind wir nur um 0,18 (0,22) Sekunden schneller. Wir dürfen aber nicht aufgeben. **Version 23** macht alles wieder wett und zwar durch den simplen Trick, daß wir das Potenzieren in seine Grundelemente zerlegen.

Sie wissen doch: 4 hoch 3 (4↑3) ist dasselbe wie »4 zweimal mit sich selbst multipliziert« (4*4*4).
25 B=4 (C entfällt)
40 A=B*B*B
Ja, da schauen Sie, gell? Beim VC 20 braucht das Programm nur 1,68, also fast 7 Sekunden weniger. Beim C 64 sind es 8,31 Sekunden, die wir sparen.

Regel 8
Die Funktion Potenzieren (↑) soll durch Mehrfach-Multiplikation ersetzt werden.

Als letztes Objekt möchte ich oft aufgerufene Unterprogramme messen. Wir erreichen das ganz einfach dadurch, daß wir das letzte Programm (Version 23) abändern. So erhalten wir **Version 24:** Die Definition der Variablen (Zeile 25) und die Multiplikation (Zeile 40) verbannen wir als Unterprogramm an das Ende des Programms und springen innerhalb der 300fachen Schleife mit GOTO darauf.

Version Nr.	Programmier-Methode	Laufzeit (Sek.) VC 20	C 64
17	Multiplikation, lang x kurz	11,85	14,15
18	Multiplikation, kurz x lang	11,48	13,71
19	Null weglassen	11,31	13,51
20	Variable vordefinieren	1,23	1,48
21	Potenzieren (4 hoch 3) mit ↑	8,81	10,53
22	Variable vordefinieren	8,63	10,31
23	4 x 4 x 4 statt 4↑3	1,68	2,01

Laufzeiten der Programmversionen für arithmetische Funktionen

Version Nr.	Programmier-Methode	VC 20	C 64
24	Unterprogramm am Ende, Sprung mit GOTO-GOTO	2,75	3,28
25	Unterprogramm am Ende, Sprung mit GOSUB-RETURN	2,61	3,13
26	Unterprogramm am Anfang, Sprung mit GOTO-GOTO	2,60	3,10
27	Unterprogramm am Anfang, Sprung mit GOSUB-RETURN	2,48	2,96

Laufzeiten der Programmversionen für Unterprogramme

25 löschen; 30 FOR Z=1 TO 300; 40 GOTO 40000
Alles andere bleibt, aber neu kommt dazu: 40000 B=4; 50000 A=B*B*B; 60000 GOTO 50

Es ist nicht weiter erstaunlich, daß dieser Umbau diese Version 24 gegenüber Version 23 verlangsamt. Aber merken Sie sich die Laufzeit, VC 20: 2,75 Sekunden, C 64: 3,28 Sekunden. Als nächstes ersetzen wir die beiden GOTO-Zeilen durch GOSUB-RETURN.
40 GOSUB 40000
60000 RETURN
Diese **Version 25** spart uns 0,14 (0,15) Sekunden. GOSUB ist schneller als GOTO!

Sie haben vielleicht schon gelesen, daß oft gebrauchte Unterprogramme am Anfang eines Programms stehen sollen. Den Grund dafür will ich Ihnen mit den nächsten zwei Versionen vorführen.

Version 26 macht das zunächst für die GOTO-Version. Wir bauen sie auf der Version 24 auf, mit folgenden Änderungen:

Die Zeitmessung lassen wir wie gehabt in den Zeilen 10, 20 und 1000, die Schleife und den Ausdruck des Resultats in den Zeilen 30, 50 und 60.

Nur beim Unterprogramm streichen wir alle Nullen der Zeilennummern, so daß es jetzt in den Zeilen 4, 5 und 6 steht. Um zu vermeiden, daß das Programm gleich mit dem Unterprogramm beginnt, fügen wir davor (Zeile 3) noch eine Umleitung ein, die sofort auf der Zeile 10 weitermacht. Schließlich brauchen wir noch den Sprung in das Unterprogramm, den wir in die Zeile 33 setzten. Das Ganze sieht jetzt so aus:

```
3 GOTO10
4 B=4
5 A=B*B*B*
6 GOTO 50
10 TI$="000000"
20 PRINT CHR$(147)
30 FOR Z=1 TO 300
33 GOTO 4
50 NEXT
60 PRINT A
1000 POKE 214,18:PRINT:PRINT TI/60 "SEKUNDEN":END
```

Nach RUN erhalten wir beim VC 20 2,6 Sekunden, beim C 64 3,1 Sekunden. Gegenüber Version 24, unserem Vergleichsobjekt, sparen wir 0,15 (0,18) Sekunden.

Dasselbe passiert, wenn wir in der **Version 27** die GOTOs mit GOSUB-RETURN ersetzen.
6 RETURN
33 GOSUB 4
Gegenüber der anderen GOSUB-Version (Version 25) sparen wir beim VC 20 0,13 Sekunden, beim C 64 0,17 Sekunden.

Regel 9
★ Der Aufruf von Unterprogrammen mit GOSUB ist schneller als mit GOTO.
★Häufig gebrauchte Unterprogramme gehören ganz an den Anfang eines Programms. Sie müssen dann allerdings zuerst mit einem GOTO umgangen werden.

Ich bin überzeugt, daß in Basic noch mehr spektakuläre Zeitgewinne stecken.

Falls Sie eine REGEL 10 oder noch mehr entdecken, ermuntere ich Sie um Mitteilung.

Wenn Sie Fragen haben, können Sie mich mit einer Leserzuschrift ansprechen.

(Dr. Helmuth Hauck/aa)



Soccer begeistert nicht nur Fußball-Fans



Viele Spiele wurden in der 64'er Redaktion angeschaut, gespielt, getestet. Eines fesselte uns immer wieder.

20.00 Uhr. Das Spitzenspiel »Bayern Albert« gegen »VC Volker 20« wird vom Schiedsrichter angepfiffen. Jetzt entscheidet sich, wer die Tabellenführung übernimmt. Anstoß von VC Volker. Langer Paß in die gegnerische Hälfte. Doch was ist das? Bayern Albert fängt den Ball ab, kontert, stürmt vor das Tor, schießt und ... der Torwart von VC Volker wirft sich in die falsche Ecke. 1:0.

Nach nervenaufreibenden drei Minuten der langerwartete Halbzeitpfiff.

Weitere drei Minuten später ist es entschieden. Die Legende um den in dieser Saison bisher ungeschlagenen »VC Volker 20« ist beendet. Der neue Tabellenführer »Bayern Albert« sitzt schweißgebadet auf seinem Stuhl und kann es noch gar nicht glauben. Es ist geschafft.

Die ersten Kommentare sind zu hören: Der Boden sei viel zu hart, der Rasen viel zu grün und das Bier für Trainer Volker viel zu warm ...

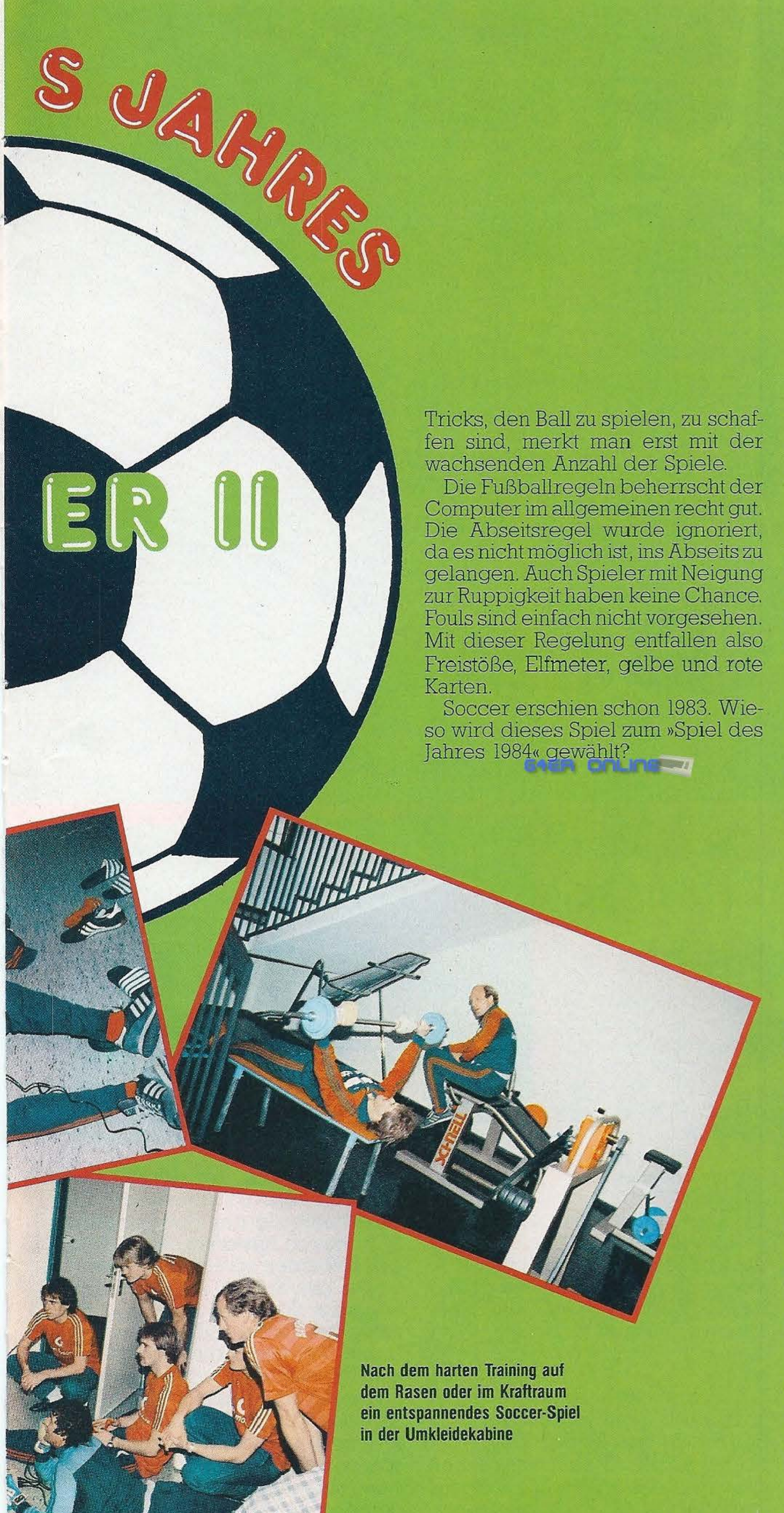
Oft trafen wir uns abends oder am Wochenende, um unsere Spieler über den Rasen laufen zu lassen. Die Soccer-Sucht hatte uns befallen.

Doch warum gerade dieses Spiel, und nicht »Summer Games«, »Pharaoh's Curse« oder eines der anderen aus der Vielzahl der Spiele für den C 64?

Soccer kann man alleine gegen den Computer oder gegen einen Partner spielen. Der Computer läßt sich auf neun Schwierigkeitsstufen einstellen. In Stufe 1 fällt es auch dem Anfänger nicht allzu schwer, den Computer zu besiegen. In der Stufe 9 muß man fast schon Profi-Kicker sein. Die Spiele gegen den Computer machen allerdings nicht halb soviel Spaß, als gegen einen zweiten Spieler.

Andere Taktik bei jedem neuen Gegner

Hier treten die Qualitäten dieses Spieles erst richtig hervor. Einfach nur nach vorne schießen, nützt in den wenigsten Fällen. Man ist gezwungen, sich Taktiken zurechtzulegen, den Gegner zu berechnen und dies, mit dem Joystick auch noch umzusetzen. Wieviele verschiedene



Nach dem harten Training auf dem Rasen oder im Kraftraum ein entspannendes Soccer-Spiel in der Umkleidekabine

Tricks, den Ball zu spielen, zu schaffen sind, merkt man erst mit der wachsenden Anzahl der Spiele.

Die Fußballregeln beherrscht der Computer im allgemeinen recht gut. Die Abseitsregel wurde ignoriert, da es nicht möglich ist, ins Abseits zu gelangen. Auch Spieler mit Neigung zur Ruppigkeit haben keine Chance. Fouls sind einfach nicht vorgesehen. Mit dieser Regelung entfallen also Freistöße, Elfmeter, gelbe und rote Karten.

Soccer erschien schon 1983. Wieso wird dieses Spiel zum »Spiel des Jahres 1984« gewählt?

Da unser Preis in diesem Jahr zum ersten Mal vergeben wird, erschien es uns fair, auch die früher veröffentlichten Programme mit einzubeziehen. Unter allen Spielen, die uns in der Redaktion erreichten (und das waren sehr viele!), hatte fast jeder der Redakteure seinen Favoriten. Aber nur von Soccer waren alle begeistert. So wählten wir gemeinsam ein »altes« Spiel.

Zum Thema Fußball gehört eine Fußballmannschaft. Für einen Fototermin konnten wir den FC Bayern München gewinnen. Wie schon auf der »Systems '83« wurde hierbei natürlich Soccer gespielt. Einstimmige Meinung: »Spitze«. Vielleicht wird Soccer in Zukunft sogar als Trainingshilfsmittel eingesetzt?

Auch die Profis waren begeistert

Der C 64 war kaum aufgebaut, da saßen die ersten Bayern-Spieler auch schon davor, obwohl sie gerade erst vom Training kamen. Dieter Hoeness und Raimund Aumann waren die ersten, die sich einen Platz am Joystick sicherten. Der Pfiff des Computerschiedsrichters lockte dann immer mehr Spieler an. Selten wird in der Kabine Fußball gespielt, doch hier waren dann fast alle dabei. Nur Lothar Matthäus, der mit der Nationalmannschaft im Trainingslager war, verpaßte diese Gelegenheit. Aber vielleicht wird auch in der Nationalmannschaft schon mit Soccer taktiert und trainiert.

Jean Marie Pfaff erzählte, daß bei ihm zuhause der C 64 fast den ganzen Tag läuft. Wenn seine Kinder nicht davor sitzen, spielt auch er sehr gerne. Natürlich nicht nur Soccer.

Soccer wird von Commodore zu einem Preis von 59 Mark angeboten. (rg)

QX 9 macht Schluß mit der plattgetretenen »Space-Invaders-Variante«.

Viele Computer-Fans sind es sicherlich leid, ständig neue Schießspiele, im altbekannten Invaders-Prinzip »vorgesetzt« zu bekommen. Nur der Feind sieht anders aus, die Grundidee bleibt die gleiche. Für diejenigen, die dennoch mit Lasern und Photonen ihre Freizeit verbringen wollen, gibt es seit kurzem für den C 64 ein Spiel, das im Happy-Software-Verlag erschienen ist. Es heißt QX 9, und basiert auf dem »Ich-sitze-im-Raumschiff-Prinzip«. Im Klartext heißt das: Der Monitor zeigt das Innere eines Raumschiffs mit Blick nach draußen. So sind bei QX 9 an beiden Seiten Instrumente angebracht, die zur Navigation des Schiffes nötig sind. QX 9 steht für ein Raumschiff im Weltraum, den eine Flotte feindlicher Raumschiffe bedroht. Letztere besteht aus einer Gruppe von Jägern und deren Mutterschiffe. Zwischen beiden schwebt ein Meteoritengürtel. Er trennt sie voneinander. Die Aufgabe des Spielers ist nun denkbar einfach: Weg damit, und zwar soviel wie möglich! Zunächst erweckt das Spiel den Anschein, als könne man nie erlernen, seinen Kreuzer sicher durch das feindliche Abwehrfeuer zu steuern; doch dies vergeht mit der Zeit, und QX 9 entwickelt sich zu einem rasanten Actionspiel. Die Grafik schwebt in der Mittelklasse, dafür ist die Sounduntermalung um so besser. Da das Spiel erst mit der Zerstörung des Schiffs endet, werden einem High-Score keinerlei Grenzen gesetzt. Anders als die anderen »Star-Raiders«-Variaten kann man mit seinem Raumkreuzer auch auf dem Satelliten aufschlagen beziehungsweise bei der Landung abstürzen. Das jedoch wertet das Spiel nur auf, und formt das Endbild zu einer gelungenen Abwechslung in der Welt der Schießspiele. Preis: 48 Mark.
(Oliver v. Quadt/aa)

Catastrophes



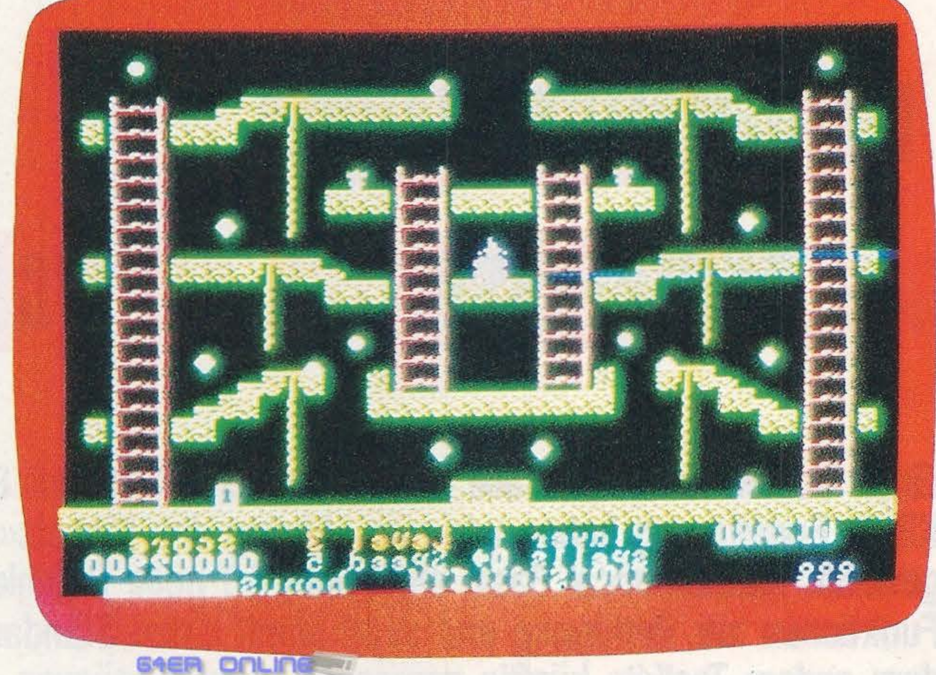
Die Grundidee ist endlich einmal nicht die eines »Miners« beziehungsweise die eines »Pac-Man«. Vielmehr geht es um ein waghalsiges Geschäft. Sie haben nämlich die Aufgabe, ein wind- und wetterfestes »Häuschen« zu bauen. Gleichzeitig müssen Sie ein bestimmtes Score-Limit erreichen und noch dazu möglichst den Gegner übertrumpfen. Nun, das ist jetzt ziemlich viel auf einmal. Zuerst jedoch zum Bildschirm: Zu erkennen sind nach der Einleitung eine schöne Meeresbucht, umgeben von hohen Bergen, ein zierlicher Leuchtturm, der später eine wichtige Rolle einnimmt, und zuletzt zwei recht unförmige, gebäudeähnliche Gebilde. Eines davon gehört Ihnen. Versuchen Sie nun, in sechs Tagen ein ansehnliches Hochhaus daraus zu machen. Steigen Sie in Ihren Hubschrauber, fliegen Sie damit zu einem Transportkutter, der geduldig am Bildrand wartet. Er hat je ein Fertigteil-Appartment geladen. Dieses muß jetzt von Ihnen zum Haus gebracht werden und schließlich richtig draufgesetzt werden. Unter »richtig« sei verstanden: hurricane-, erdbeben-, gewitter- und überschwemmungsfest! Eine gewiß nicht leichte Aufgabe, doch das ist der Reiz an dem Spiel. Nach jedem Tag (man sieht die Sonne richtig wandern) wird eine neue Mindestpunktzahl festgesetzt. Wird sie nicht erreicht, scheidet der Spieler aus.

Am Ende des sechsten Tages erscheint der »Daily Mirror«, der über den Erfolg des Bauvorhabens berichtet. Große Pluspunkte erzielt das Spiel, das von Happy-Software vertrieben wird, wegen der originellen Grundidee und deren Ausführung. Grafik und Musik sind ebenfalls sehr gut. Besonders die Unterbrechung des Spielvorgangs ist positiv zu bewerten: Nach wenigen Sekunden springt der Computer für den Spieler ein. Die leider englische Anleitung im Programm kann auf- beziehungsweise abgescrollt werden. Grundsätzlich formen die einzelnen Reize des Spiels und seine Finessen im Detail »Catastrophes« (für C 64: Preis: 48 Mark) zu einem lustigen, aber auch sehr spannenden Geschicklichkeitsspiel, das sogar dem actionliebenden Arcade-Fan viel Spaß machen wird.
(Oliver v. Quadt/aa)

Wizard

ein würdiger Nachfolger für Jumpman

Mit Wizard hat die amerikanische Softwarefirma PP & S (Progressive Peripherals and Software) ein Spiel auf den Markt gebracht, das den Rennern Jumpman und Jumpman Junior in nichts nachsteht.



Einer der 39 Levels von Wizard

Wizard ist nicht etwa, wie der Name vermuten läßt, ein Zaubererspiel, sondern ist eher in die Kategorie der intelligenten »Jump- and Run-Spiele« einzuordnen. Intelligent deshalb, weil man auch selbst kreativ das Spiel gestalten kann.

Das Spiel

Das Ziel des Spieles ist es, möglichst viele Punkte zu sammeln. Diese erhält man, indem man möglichst viele Schätze sammelt, oder möglichst schnell das Ende des Levels erreicht. Das Ende einer Spielstufe erreicht man dann, wenn man es geschafft hat, den Schlüssel zu ergattern und damit das Schlüsselloch (also den Ausgang) zu erreichen. Daß dies nicht so einfach ist, wird sich wohl jeder denken können: Schließlich hat man mit 20 verschiedenen Arten von Feinden zu rechnen, und mit der Gefahr, irgendwo zwischen den vielen Leitern, Seilen, Treppen und gefährlichen Feuern abzustürzen. Weitere Gemeinheiten wie Teleporter und Transporter sind in höheren Levels in immer stärkerem Maße eingebaut.

Hätte man da nicht noch so tolle Dinge wie Zaubersprüche, dann wäre man spätestens nach der 15. Spielstufe dem Tod geweiht. Bei sinnvoller Nutzung der zur Verfügung stehenden Sprüche kann man den letzten, also den 39. Level mit ein bißchen Übung erreichen.

Keine Sorge, auch nachdem man den letzten Schwierigkeitsgrad geschafft hat, wird das Spiel nicht so schnell langweilig: Wer will, kann sich seine eigenen Levels erstellen.

Das »Construction Kit«

Bei der Entwicklung eigener Bilder steht dem »Bildschirmkünstler« die Wahl zwischen fünf Menüpunkten offen: Die Wahl der Monster, die Zaubersprüche, das Editieren des Bildschirms, die Startposition des »Wizard« und die Farbwahl. Der Bildschirmeditor besteht aus einer Kombination von Cursorbewegung per Joystick und Gegenstandsauswahl durch zwei Tasten. Hat man den Bildschirmeditor verlassen, kann man beginnen, die ersten Bosheiten einzubauen. Zur Wahl stehen 128 verschiedene Sprites, deren Auftrittshäufigkeit und Animationsgeschwindigkeit durch den Spieler bestimmt werden können. Falls Sie sich zu viele »Shadow Lords«, Riesenratten oder »Slimes« eingebaut haben, besteht noch immer die Möglichkeit, den Level durch Zaubersprüche (Unsichtbarkeit, Teleport, Feuerball etc.) zu erleichtern.

Auf eine Diskettenseite passen insgesamt 100 Levels. Wer sich nicht sattspielen kann, braucht sich also deswegen nicht gleich massenweise Disketten kaufen.

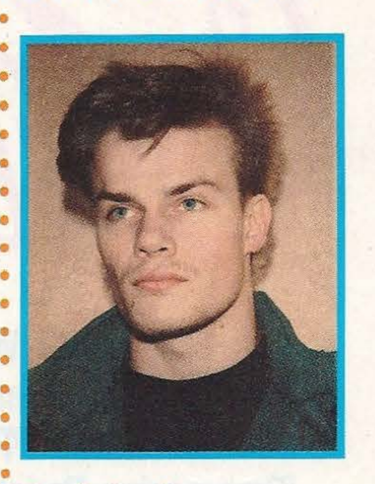
Eine Fortsetzung ist geplant

Wizard ist ein Spiel, das durchaus mit Hüpfspielklassikern wie Jumpman konkurrieren kann, und das zu kaufen es sich lohnt.

Übrigens: Die Herstellerfirma PP & S plant für Anfang 1985 ein »Wizard Expansion Set« mit erweiterten Möglichkeiten und 40 neuen, gut durchdachten Levels. Einem Erfolg dieses Spiels dürfte eigentlich nichts mehr im Wege stehen.

(Manfred Kohlen/aa)

MATHEMATICAL-BASIC: DAS SUPER-BASIC FÜR DEN VC 20



Der Autor des »Mathematical Basic« stellt sich vor

Ich bin am 9.7.1963 geboren und zur Zeit Praktikant in einem Softwarehaus in Bad Kreuznach.

Für die Entwicklung des Programms ist hauptsächlich der Ärger über das permanente Fehlen der gebräuchlichsten naturwissenschaftlichen Funktionen in jedem Basic-Interpreter von Commodore verantwortlich.

Um mehrere Funktionen über den USR-Vektor zu implementieren, müßte man vor jedem Aufruf auch lästigerweise die jeweilige Adresse in diesen Vektor schreiben. Das ist sehr umständlich; also müssen mehrere USR-Funktionen her. Wenn nun aber schon eine Routine für das Selektieren der Adressen geschrieben wird, so kann man auch gleich das Ganze etwas komfortabler gestalten und eine Interpreter-Erweiterung basteln. Das waren meine anfänglichen Überlegungen und heraus kam »Mathematical Basic«. Das Programm wurde so gehalten, daß es jederzeit weitere Befehle und Funktionen aufnehmen kann. 6502-Freunde mit Kenntnissen über das VC 20-Betriebssystem werden sich schnell mit »Mathematical Basic« zurechtfinden und auch eigene Änderungen durchführen können.

(Wolfgang W. Wirth)

Einen echten Knüller für alle Besitzer eines VC 20 mit 8 KByte Erweiterung haben wir zum »Listing des Monats« gewählt. Mathematical Basic stellt über 50 neue Befehle und Funktionen zur Verfügung und setzt damit einen Standard, an dem andere Toolkits künftig gemessen werden müssen.

Das Programm ist vollständig in Assembler geschrieben und liegt zum Eintippen als Basic-Lader vor. Nach dem Starten des Laders wird das eigentliche Maschinenprogramm ab Adresse 29211, also am oberen Ende von Speicherblock 3 abgelegt und gleichzeitig vor dem Überschreiben durch Basic-Programme oder Variable geschützt. Bei einem voll ausgebauten VC 20 (+ 24 KByte) kann das Programm ohne weitere Veränderungen benutzt werden. Aber auch Besitzer einer 8-KByte-Erweiterung können diese Basic-Erweiterung nutzen. Allerdings muß dann diese 8-KByte-Erweiterung hardwaremäßig auf den Adreßbereich von $6000 bis $7FFF, also auf Speicherblock 3 umgeschaltet werden. Schlagen Sie bitte die erforderliche Vorgehensweise im Handbuch für die 8-KByte-Erweiterung nach. Es ist die gleiche Umstellung, die auch nötig wird, wenn Sie das 8-KByte-Modul zusammen mit einer 16-KByte-Erweiterung als 24 KByte RAM nutzen wollen.

Haben Sie allerdings keine zusätzliche 16-KByte-Erweiterung, dann steht Ihnen für das Arbeiten mit Mathematical-Basic nur der Grundversionsspeicher zur Verfügung. Leider ist es nicht möglich, nur das 16-KByte-Modul zu verwenden, da dieses immer auf Speicherblock 1 und 2 eingestellt ist. Doch nun zum Programm selbst: Mathematical-Basic erweitert den Wortschatz des Basic-Interpreters um zusätzliche Befehle und Funktionen. Insbesondere wurde die Handhabung der trigonometrischen Funktionen verbessert (man kann zwischen Altgrad, Neugrad und Bogenmaß wählen). Daneben wurden Spezialbefehle für Diskettenbetrieb wie CATALOG, DLOAD, DSAVE etc. eingebaut. Strukturierte Programmierung wird durch DO...UNTIL-Schleifen unterstützt. Viele weitere Befehle und Funktionen vereinfachen die Programmierung zum Teil ganz erheblich.

Auf jeden Fall braucht »Mathematical-Basic« den Vergleich auch mit kommerziellen Softwareprodukten dieser Art nicht zu scheuen.

(ev)

Musik, Musik, Musik

Das Programm ist als Werkzeug zum Experimentieren mit dem hardwaremäßig phantastischen aber softwaremäßig völlig vernachlässigten »Sound Interface Device« (SID) im C 64 gedacht. Nicht zuletzt kann man mit dem Synthesizer auch musizieren!

Um die Möglichkeiten zu nutzen, sind folgende Parameter einstellbar und werden übersichtlich — teilweise grafisch unterstützt — auf dem Bildschirm dargestellt.

1) Die Tastenreihe »Q« bis »RETURN« bildet die Orgeltastatur. Sie umfaßt zwei Oktaven mit allen Halbtönen und erscheint oben auf dem Bildschirm. Beim Drücken einer Taste erklingt der Ton so lange, bis sie wieder losgelassen wird. Gleichzeitig zeigt ein gelber Balken (Sprite) die aktivierte Orgeltaste an.

2) Anschlag (Attack), Abschwellen (Decay), Haltepegel (Sustain Level) und Ausklingen (Release) sind mit den Funktionstasten einstellbar. Die geshifteten Tasten verringern, die ungeshifteten erhöhen den Wert. Die relative Lautstärke (Pegel) des Tones wird ständig rechts oben als Balken auf dem Bildschirm gezeigt.

3) Als Wellenformen kann man Dreieck, Sägezahn, Rechteck, Rauschen und die ringmodulierte Dreieckschwingung (RING) mit der Taste »Z« wählen. Die beiden letzten sind in Klammern eingefaßt, da sie sich zum Musizieren nicht eignen.

4) Wählt man die Rechteckschwingung, wird das Tastverhältnis angezeigt und kann mit den Tasten »N« und »M« in Sechzehntel-Abstufungen verändert werden. Das Low-Byte des Tastverhältnisses beträgt immer 128.

5) Der Synthesizer nutzt das im Anhang P des Handbuchs gezeigte Frequenzspektrum voll aus: Der tiefste Ton ist der Halbton unter C_0 und der höchste ist das Ais-7 (!).

Da die Tastatur aber nur zwei Oktaven umfaßt, kann man mit den Cursortasten den Frequenzbereich um jeweils eine Oktave ändern.

6) Mit der Taste »C« wird der Filtermodus umgeschaltet, wobei das entsprechende Wort auf dem Bildschirm revers geschrieben wird. Ist kein Filtermodus eingeschaltet, werden die Schaltbits 0, 1 und 2 im Register 23 gelöscht, also für alle drei Stimmen die Filter ausgeschaltet. Auch die Verwendung mehrerer Filtermodi ist zugelassen.

7) Die folgenden Parameter sind nur bei eingeschalteten Filtern hör- und sichtbar:

7a) Mit den (ungeshifteten) Tasten »[« und »]« verändert man die Grenzfrequenz der Filter in Sechzehntelschritten. Es werden also die höchstwertigen vier Bits im Register 22 variiert.

7b) Die Filterresonanz wird mit den Tasten »<« und »>« vergrößert oder verkleinert.

7c) Die Tasten »?« beziehungsweise ».« schalten den »Wahwah«-Effekt (er heißt wie er klingt) ein und aus. Nach dem Einschalten wird die Filtergrenzfrequenz (Register 22) gemäß dem Verlauf der Hüllkurve (Register 28) verändert. Die eingestellte Filterfrequenz (siehe 7a) hat dann keine Bedeutung.

8) Mit der Stopp-Taste verläßt man das Programm. Da es jetzt im Speicher ist, kann es mit RUN 1000 unverzüglich wieder gestartet werden. Nun haben alle Parameter wieder die ursprünglichen Werte.

Ich möchte nicht verheimlichen, daß der »Synthesizer« natürlich auch einige Einschränkungen hat:

1) Alle Parameter werden jeweils für alle drei Stimmen gleichzeitig eingestellt, denn bei nur einem Manual wäre es sinnlos, den verschiedenen Stimmen zum Beispiel verschiedene Wellenformen zu geben.

2) Es gibt natürlich grenzenlos viele Möglichkeiten, Parameter (zum Beispiel die Filterfrequenz) während des Erklingens eines Tones zu verändern. Ihr Einbau würde aber ebenfalls den Rahmen dieses Programms sprengen.

3) Die Lautstärke ist fest auf 15 eingestellt. (Wozu hat man schließlich den Regler am Fernseher?)

4) Eine Computertastatur ist kein Orgel-Keyboard. Dies setzt dem Spieler von Musikstücken natürlich Grenzen. Für Bastler dürfte es aber nicht allzu schwierig sein, parallel zur Tastatur ein in der richtigen Matrix verdrahtetes Keyboard anzuschließen.



Der Bildschirm zeigt alle Werte auf einen Blick

Charakter	Okt.	Wellenf.	Tastv.	An	Ab	Hlt	Aus	Filt	Wah	FF	Res
E-Gitarre	4-6	Rechteck	1	0	9	0	9	HBT	Ein	-	7
Mandoline	5-7	Rechteck	C	0	8	00	8	-	-	-	-
Mundharmonika	6-8	Sägezahn	-	8	0	F	7	-	-	-	-
Gong	3-5	Ring	-	0	D	0	D	-	-	-	-
Explosion	0-2	Rauschen	-	0	0	F	C	-	-	-	-
Popcorn	5-7	Dreieck	-	0	2	0	0	-	-	-	-
Quak-quak	4-6	Sägezahn	-	8	9	9	8	B	Ein	-	F
Meeresrauschen	5-7	Rauschen	-	B	B	0	B	HBT	Ein	-	7
Knattern	0-2	Rechteck	0	8	0	F	D	-	-	-	-
Xylophon	4-6	Dreieck	-	0	9	0	9	T	Aus	2	6
Glöckchen	6-8	Dreieck	-	0	A	0	A	-	-	-	-

Beispiele für Klangeinstellungen mit »Synthesizer«

Abspeichern des Programms

Wer es ganz elegant machen möchte, kann das Maschinenprogramm (den »Objektcode«) als solches auf Band oder Diskette schreiben. Ich habe die dazu nötigen POKEs und SYS-Aufrufe zusammengestellt, die man im Direktmodus eingeben kann. Allerdings sind die PRINT-Befehle (Zeilen 1000-1200)



nicht im Maschinenprogramm enthalten! Aus diesen Zeilen sollte man also ein kleines Basic-Programm schreiben, das mit der Zeile »LOAD "SYNTHESIZER-OBS"«, Gerätenummer, 1« beginnt und mit »SYS12800« endet.

Wer einen Monitor besitzt, kann natürlich diesen zum Abspeichern benutzen. Die Startadresse ist $ 2F00 und die Endadresse $ 35DD.

Nun noch ein Hinweis für diejenigen, die den SID selbst programmieren wollen und nur das Handbuch zum Commodore 64 besitzen: Wer die Beispielprogramme schon ausprobiert hat, dem sollte aufgefallen sein, daß die Töne nie ausklingen — egal, welchen Wert man hierfür eingestellt hat. Das liegt daran, daß der Tongenerator dort immer mit POKE W, 0 »abgewürgt« wird. Zum korrekten Ausschalten eines Tones darf man aber nur das 0. Bit des Wellenform-Registers löschen, das sogenannte »KEY«-Bit! Richtig heißt es also POKE W, 16 (bei der Dreieckschwingung).

(Martin Ahlborn/rg)

aa	=	Anfangsadresse
ea	=	Endadresse
tn	=	Ton (Frequenzparameter)
hb%	=	Hi-Byte von tn
lb%	=	Lo-Byte von tn
hx$	=	Hexadezimalzahl
ps	=	Prüfsumme
h	=	Hi-Nibble der Hexzahl hx$
l	=	Lo-Nibble der Hexzahl hx$
Funktion		
fn rd (x)	=	x auf ganze Zahl runden (eine nützliche Funktion, auch wenn sie hier nicht unbedingt notwendig ist)

Variablenliste des Basic-Ladeprogrammes »Synthesizer«

20 — 50	Schreiben des Maschinenprogramms und der dafür nötigen drei Tabellen in den Speicher.
100 — 190	Hier wird die im Anfang P des Handbuchs gezeigte Tabelle rechnerisch erstellt. Sie besteht aus 96 Frequenzparametern, die in Lo- und Hi-Byte zerlegt werden. Sie wachsen logarithmisch, das heißt, der Faktor zwischen zwei Zahlen ist konstant: 12te Wurzel aus 2 (eine Oktave hat 12 Halbtöne).
300 — 350	Die Prüfsumme wird mit dem Sollwert verglichen. Wenn sie stimmt, startet das Programm.
700 — 750	POKE-Routine: Die Hexadezimalzahlen in den DATAs werden gelesen, nach dezimal gewandelt und in den Speicher geschrieben. Außerdem wird eine Prüfsumme gebildet.
1000 — 1200	Das Bild wird mit Basic erstellt, was gegenüber der Maschinensprache (ausnahmsweise) keinen spürbaren Geschwindigkeitsverlust bringt. Die PRINT-Texte müssen genau abgeschrieben werden, da sie vom Maschinenprogramm teilweise geändert werden!
1310	Aufruf des Maschinenprogramms.
1500 — 1750	Drei Tabellen: Die erste enthält die Koordinaten für die Sprites, die die gedrückten Orgeltasten anzeigen. Die zweite Tabelle liefert die für die Tastaturabfrage erforderlichen Bitmuster der Ports A und B des CIA 1. (Dies ist deswegen so umständlich, weil auch mehrere gleichzeitig gedrückte Tasten jeweils eindeutig erkannt werden müssen). Die dritte Tabelle enthält die Bildschirmcodes einiger Wörter, die bei Tastendruck erscheinen.
2000 — 3200	Das 989 Bytes lange Maschinenprogramm.

Die Zeilen 300, 350 und 750 können entfernt werden, nachdem die DATA-Zeilen einmal richtig abgeschrieben wurden.

Programmbeschreibung »Synthesizer«

Listing »Synthesizer«

```
1 rem ------- c-64 synthesizer -------
2 rem    von martin ahlborn        4/5 1984
3 rem        hohe feldstrasse 1
4 rem        3418 uslar 2
5 rem
10 print"▒▒▒▒"spc(12)"bitte warten"
20 aa=12288:ea=12339:gosub700
30 aa=12352:ea=12403:gosub700
40 aa=12465:ea=12520:gosub700
50 aa=12800:ea=13788:gosub700
88 :
90 rem *************************
92 rem *notentabelle erstellen*
94 rem *************************
96 :
100 deffn rd(a)=int(a)-((a-int(a))>=.5):
 rem auf ganze zahl runden
110 tn=67284:rem parameter fuer h-7
120 forz=95to0step-1
130 tn=tn/2↑(1/12)
140 tn=fn rd(tn)
150 hb%=tn/256
160 lb%=tn-hb%*256
170 poke12032+2*z,lb%
180 poke12032+2*z+1,hb%
190 next
290 :
300 ifps<>15145then350
320 goto1000
350 print"falsche daten !":end
688 :
690 rem **************
692 rem *poke-routine*
694 rem **************
696 :
700 forpc=aatoea
710 readhx$
720 h=asc(hx$)-48:h=h+(h>9)*7
730 l=asc(right$(hx$,1))-48:l=l+(l>9)*7
740 pokepc,16*h+l
750 ps=ps+h+l :rem pruefsumme
760 next:return
980 :
990 rem ***************
992 rem *text & grafik*
994 rem ***************
996 :
1000 printchr$(142)"▒▒▒▒********  c-64
synthesizer  ********▒"
1010 print"▒ ▒ ▒ ▒ | ▒ ▒ ▒ | ▒ ▒ | ▒ ▒ ▒
 | ▒   max ▒"
1020 print" ▒ ▒ ▒ | ▒ ▒ ▒ | ▒ ▒ | ▒ ▒ ▒
| ▒▒▒▒▒▒▒ "
1030 print" ▒ ▒ ▒ | ▒ ▒ ▒ | ▒ ▒ | ▒ ▒ ▒
|r▒ pegel ▒ "
1040 print" ▒ | | | | | | | | | | | | |
|e▒▒▒▒▒▒▒ "
1050 print" ▒q |w |e |r |t |y |u |i |o |p |@ |* |↑ |=
|t▒   min ▒"
1060 print"▒▒▒▒tasten     parameter     01
23456789abcdef";
1070 print" ____________________________________

____________";
1080 print"▒ f1  f2     anschlag        ▒LLL
LLL▒           ";
1090 print"▒ f3  f4     abschwellen  ▒L▒
               ";
1100 print"▒ f5  f6     halten          ▒LLL
```

```
LLLLLLLLLLLLL▒";
1110 print"▒ f7  f8    ausklingen   ▒LLL
LLLL▒          ";
1120 print"▒ 'n' 'm'   tastverh.    ▒LLL
LLL▒           ";
1130 print"▒  'z'      wellenform   drei
eck   "
1140 print"▒  crsr     oktaven      c-▒4
▒ bis c-▒6"
1150 print"▒  'c'      filter       hoch
 band tief"
1160 print"▒ '[' ']'   filterfreq.  ▒LL
LLLL▒          ";
1170 print"▒ '<' '>'   resonanz     ▒LLL
LLLL▒          ";
1180 print"▒  '?'      'wah-wah'    aus"
1190 print"▒ run/stop  ende"
1200 print"▒▒(c) 1984      martin ahlbor
n 3418 uslar▒"
1300 :
1305 rem ************
1310 ::::::sys12800::
1320 rem ************
1480 :
1490 rem **********
1492 rem *tabellen*
1494 rem **********
1496 :
1500 data10,40,18,50,20,40,28,50
1510 data30,40,38,50,48,50,50,40
1520 data58,50,60,40,68,50,70,40
1530 data78,50,88,50,90,40,98,50
1540 dataa0,40,a8,50,b8,50,c0,40
1550 datac8,50,d0,40,d8,50,e0,40
1560 datae8,50,f8,50
1590 :
1600 data7f,01,7f,40,7f,08,fd,02
1610 datafd,01,fd,40,fb,02,fb,01
1620 datafb,40,fb,08,f7,02,f7,01
1630 dataf7,40,ef,02,ef,01,ef,40
1640 dataef,08,df,02,df,40,df,08
1650 databf,02,bf,01,bf,40,bf,08
1660 databf,20,fe,02
1690 :
1700 data28,12,01,15,13,03,08,05
1710 data0e,29,12,05,03,08,14,05
1720 data03,0b,20,20,13,01,05,07
1730 data05,1a,01,08,0e,20,04,12
1740 data05,09,05,03,0b,20,20,20
1750 data28,12,09,0e,07,29,20,20
1760 data20,20,05,09,0e,01,15,13
1980 :
1990 rem **********
1992 rem *programm*
1994 rem **********
1996 :
2000 datad8,a9,00,a2,7e,9d,80,03
2010 dataca,10,fa,a9,ff,a2,24,9d
2020 datac1,03,ca,ca,ca,d0,f8,8d
2030 data81,03,a2,03,a0,00,8c,20
2040 datad0,8c,21,d0,8c,e9,07,8c
2050 dataf4,07,8c,2a,d0,a9,0e,8d
2060 datafb,07,a2,08,8e,10,d0,ca
2070 data8e,27,d0,8e,28,d0,8e,29
2080 datad0,8e,17,d0,a9,60,85,fa
2090 data8d,f2,07,8d,f1,07,8d,16
2100 datad4,8d,17,d4,a9,34,8d,f3
2110 data07,a9,02,8d,15,d4,a9,2f
2120 data85,fb,a9,80,8d,8a,02,8d
2130 data02,d4,8d,09,d4,8d,10,d4
2140 dataa2,10,8e,ee,07,a9,39,8d
2150 data06,d0,ca,8e,f8,07,8e,f9
2160 data07,8e,fa,07,8e,18,d4,8e
2170 dataef,07,a9,05,8d,e8,07,8d
2180 dataec,07,a9,0f,8d,ea,07,a9
2190 data06,8d,eb,07,20,54,34,ac
2200 dataee,07,8c,04,d4,8c,0b,d4
2210 data8c,12,d4,a9,08,8d,15,d0
2220 data10,0b,aa,a9,cc,9d,a8,05
2230 dataa9,20,9d,a9,05,ad,1c,d4
2240 dataac,f4,07,f0,03,8d,16,d4
2250 data4a,4a,4a,85,02,38,a9,64
2260 datae5,02,8d,07,d0,a6,cb,e0
2270 data40,f0,c4,78,a0,00,ba,86
2280 data02,b9,40,30,8d,00,dc,ad
2290 data01,dc,39,41,30,d0,03,4c
2300 data11,34,c8,c8,c0,34,d0,e9
2310 databa,e4,02,f0,03,4c,87,34
2320 data58,20,3e,f1,c9,03,d0,03
2330 data4c,c3,35,c9,43,d0,03,4c
2340 data55,35,c9,3b,d0,05,a2,00
2350 data4c,a5,33,c9,3a,d0,05,a2
2360 data00,4c,bc,33,c9,2e,d0,05
2370 dataa2,01,4c,a5,33,c9,2c,d0
2380 data05,a2,01,4c,bc,33,c9,2f
2390 datad0,03,4c,30,34,c9,1d,d0
2400 data03,4c,e7,33,c9,11,d0,03
2410 data4c,f7,33,c9,5a,d0,03,4c
2420 dataf3,34,c9,4d,d0,04,a2,04
2430 data10,1b,c9,4e,d0,04,a2,04
2440 data10,27,c9,85,90,04,c9,8d
2450 data90,03,4c,bd,32,38,e9,85
2460 datac9,04,b0,11,aa,bd,e8,07
2470 datac9,0f,f0,ee,38,69,00,9d
2480 datae8,07,4c,97,33,38,e9,04
2490 dataaa,bd,e8,07,d0,03,4c,bd
2500 data32,38,e9,01,9d,e8,07,48
2510 data20,54,34,68,18,69,28,ca
2520 data10,fa,4c,b2,32,bd,f1,07
2530 data29,f0,c9,f0,f0,36,bd,f1
2540 data07,18,69,10,9d,f1,07,9d
2550 data16,d4,d0,13,bd,f1,07,29
2560 dataf0,f0,21,bd,f1,07,38,e9
2570 data10,9d,f1,07,9d,16,d4,4a
2580 data4a,4a,4a,ca,d0,03,18,69
2590 data28,aa,a9,cc,9d,10,07,a9
2600 data20,9d,11,07,4c,bd,32,a5
2610 datafa,c9,90,f0,21,18,69,18
2620 data85,fa,ee,f3,07,d0,0c,a5
2630 datafa,f0,13,38,e9,18,85,fa
2640 datace,f3,07,ae,f3,07,8e,c2
2650 data06,e8,e8,8e,ca,06,4c,bd
2660 data32,ba,8a,18,69,0c,c5,02
2670 datad0,03,4c,f2,32,b9,00,30
2680 data48,b9,01,30,48,b1,fa,48
2690 datac8,b1,fa,48,88,4c,f2,32
2700 dataad,f4,07,a0,02,49,01,8d
2710 dataf4,07,f0,0c,b9,e3,30,99
2720 data60,07,88,10,f7,4c,bd,32
2730 datab9,e6,30,99,60,07,88,10
2740 dataf7,4c,bd,32,ad,e8,07,0a
2750 data0a,0a,0a,0d,e9,07,8d,05
2760 datad4,8d,0c,d4,8d,13,d4,ad
2770 dataea,07,0a,0a,0a,0a,0d,eb
2780 data07,8d,06,d4,8d,0d,d4,8d
2790 data14,d4,ad,ec,07,8d,03,d4
2800 data8d,0a,d4,8d,11,d4,60,ad
2810 dataf4,07,d0,06,ad,f1,07,8d
```

Listing »Synthesizer«

```
2820 data16,d4,68,8d,0f,d4,68,8d
2830 data0e,d4,ac,ee,07,c8,8c,12
2840 datad4,68,8d,01,d0,68,8d,00
2850 datad0,a9,09,8d,15,d0,ba,e4
2860 data02,f0,3d,68,8d,08,d4,68
2870 data8d,07,d4,ac,ee,07,c8,8c
2880 data0b,d4,68,8d,03,d0,68,8d
2890 data02,d0,a9,0b,8d,15,d0,ba
2900 datae4,02,f0,1c,68,8d,01,d4
2910 data68,8d,00,d4,ac,ee,07,c8
2920 data8c,04,d4,68,8d,05,d0,68
2930 data8d,04,d0,a9,0f,8d,15,d0
2940 data4c,00,33,a2,a6,86,fc,a2
2950 data30,86,fd,ad,ee,07,29,04
2960 dataf0,09,a9,10,8d,ee,07,85
2970 data02,10,2f,ad,ee,07,29,70
2980 datad0,0d,a9,d8,85,fc,a9,14
2990 data8d,ee,07,a0,0a,10,2b,ad
3000 dataee,07,0a,8d,ee,07,85,02
3010 datac9,40,f0,04,a9,00,f0,02
3020 dataa9,0a,a2,27,9d,58,da,ca
3030 data10,fa,a5,fc,18,69,0a,85
3040 datafc,a5,02,0a,85,02,90,f2
3050 dataa0,0a,b1,fc,99,97,06,88
3060 datad0,f8,4c,bd,32,a2,0d,bd
3070 datae8,06,29,7f,9d,e8,06,ca
3080 data10,f5,ae,ef,07,e0,0f,d0
3090 data03,20,ab,35,e0,7f,d0,05
3100 data20,ab,35,a2,ff,8a,18,69
3110 data10,8d,ef,07,8d,18,d4,0a
3120 data85,02,a2,06,86,fd,a2,e8
3130 data86,fc,a2,03,a5,02,0a,85
3140 data02,90,0b,a0,03,b1,fc,09
3150 data80,91,fc,88,10,f7,a5,fc
3160 data18,69,05,85,fc,ca,d0,e4
3170 data4c,bd,32,ad,f2,07,49,07
3180 data8d,f2,07,8d,17,d4,a0,6e
3190 datab9,f8,da,49,0e,99,f8,da
3200 data88,10,f5,60,4d,41,52,54
3210 data49,4e,20,41,48,4c,42,4f
3220 data52,4e,20,33,34,31,38,20
3230 data55,53,4c,41,52
4000 ende
ready.
```

Listing »Synthesizer« (Schluß)

```
10 *save-routine fur c-64 synthesizer
20 *bitte im direktmodus eingeben,
30 *also ohne zeilennummern
40 *
100 poke 781,geraetenummer (1 bzw 8)
110 sys 65466
120 nm$="synthesizer-obj"
130 poke183,len(nm$)
140 poke781,681and255
150 poke782,681/256
160 forc=1to15:poke680+c,asc(mid$(nm$,c)
):next
170 poke250,12032and255
180 poke251,12032/256          *startadresse
190 poke780,250
200 poke781,13790and255
210 poke782,13790/256          *endadresse
220 sys65496                   *save
ready.
```

Listing »Save-Routine« für »Synthesizer«

Mathematical Basic

Fortsetzung von Seite 50

Eine gute Nachricht für alle Freunde des VC 20: Unser »Listing des Monats« macht mit über 50 neuen Befehlen das Programmieren zum Vergnügen.

Leider gibt es aber auch eine schlechte Nachricht. Sie müssen gut und gerne 150 DATA-Zeilen eintippen. Lassen Sie sich aber davon nicht entmutigen. Der Aufwand lohnt sich ganz bestimmt. Das Programm hilft Ihnen bei der wohl unvermeidlichen Suche nach Tippfehlern durch das blockweise Bilden von Prüfsummen.

Befehle des Mathematical Basic

';:
erlaubt eine REM-Anweisung innerhalb einer Zeile ohne sofortige Programmfortsetzung ab folgender Zeile.

BEEP h, l:
gibt einen Ton variabler Höhe (h) und Länge (l) aus. h ist im Bereich von 0 bis 126 und l von 0 bis 255 definiert. Beträgt l=0, so wird kein Ton ausgegeben.

CATALOG:
bringt das Inhaltsverzeichnis der Diskette auf den Bildschirm. Dieser Befehl ist nur für das Gerät mit der Nummer 8 definiert. Wird während der Ausgabe die STOP-Taste betätigt, so wird die Ausgabe sofort abgebrochen. Wird die Leertaste gedrückt, so wird die Ausgabe nur unterbrochen und kann mit einer weiteren Betätigung dieser Taste fortgesetzt werden.

COLOR
c,h,r: setzt die Farben für Cursor (c), Hintergrund (h) und Rahmen (r). r und c erstrecken sich von 0 bis 7 und h von 0 bis 15. Die einzelnen Farben bezüglich der Nummercodes können dem VC 20-Handbuch entnommen werden.

CLS:
löscht den Bildschirmspeicher.

DEFUSRn TO x:
definiert einen der neun möglichen USR-Vektoren. Die einzelnen USR-Funktionen werden durch das Zeichen n unterschieden. n kann dabei A, B, C, D, E, F, G oder H sein. Die neunte USR-Funktion ist die, welche über den Vektor in den Adressen 1 und 2 angesprungen wird. Soll dieser Vektor definiert werden, so muß n weggelassen werden. Die Variable x steht für den Vektor selbst, also der Adresse, ab der die USR-Funktion starten soll. Beispiel: DEFUSRA TO 30000. Die USR-Funktion USRA erhält die Einsprungadresse 30000.

DEGREE:
stellt die Routinen für trigonometrische Funktionen auf Normalgrad (0 bis 360) ein.

DELETE a-b:
löscht Zeilen eines Basic-Programms von Zeile a bis Zeile b. Für a und b sind nur Konstanten erlaubt.

DIRECTORY:
siehe CATALOG

DLOAD pn$:
lädt ein Programm mit dem Namen pn$ vom Floppy-Laufwerk mit der Nummer 8. Außer dem Namen sind keine weiteren Parameter erlaubt.

DO:
erlaubt zusammen mit UNTIL eine Schleife. Der zwischen diesen beiden Schlüsselwörtern liegende Programmteil wird so oft wiederholt, bis der Ausdruck, der UNTIL folgt, nicht mehr auf logisch 0 ist. Dazu ein Beispiel: DO GET a$: UNTIL a$=" ". Das Programm verläßt die Schleife nicht eher, bis die Leertaste gedrückt wird. DO-UNTIL-Schleifen können acht Ebenen

tief geschachtelt werden. Eine neunte DO-Anweisung würde die Meldung »out of memory« zur Folge haben. Sollte UNTIL einmal ohne einen vorangegangenen DO-Befehl aufgerufen werden, so gibt der VC 20 »until without do« aus.

DSAVE pn$:
sichert ein Programm mit dem Namen pn$ auf dem Floppy-Laufwerk mit der Nummer 8. Außer dem Namen sind keine weiteren Parameter zulässig. Sollte dem Befehl DSAVE, DLOAD oder DVERIFY kein Parameter folgen, so wird automatisch »*« als Parameter gesetzt.

DVERIFY pn$:
vergleicht das Programm mit dem Namen pn$ im Speicher mit dem gleichen auf der Diskette. Als Parameter ist nur der Name erlaubt.

EXECUTE a$:
wandelt den String a$ in Interpretercode und führt die darin enthaltenen Anweisungen aus. Beispiel: a$="a=22":EXECUTE a$. Der Variablen a wird der Wert 22 zugewiesen. Der Befehl darf nicht im Direkt-Modus gegeben werden und der Befehlsstring (a$) darf eine Länge von 88 Zeichen nicht überschreiten.

GRAD:
stellt die trigonometrischen Routinen auf Neugrad (0 bis 400).

IF:
Die IF...THEN-Anweisung ist in ihren Aufbaumöglichkeiten erweitert worden. Es ist nicht mehr zwingend erforderlich, den THEN-Dummy zu schreiben. Beispiel: IF A=4 THEN PRINT B$ läßt sich auch als IF A=4 PRINT B$ eingeben.

INITIALIZE:
initialisiert die Floppy mit der Nummer 8.

LOCATE z,s:
setzt den Cursor in Zeile z und Spalte s. z liegt im Bereich von 1 bis 23 und s von 1 bis 22. Folgende Variationsmöglichkeiten sind gegeben: LOCATE z definiert nur eine neue Zeilenposition. LOCATE ,s setzt den Cursor innerhalb einer Zeile nur an eine neue Spaltenposition.

LPRINT:
verhält sich wie PRINT. Die Zeichen werden aber nicht auf den Bildschirm, sondern auf den Drucker (Nummer 4) ausgegeben. LPRINT und PPRINT sollten nur bei angeschlossenen Geräten benutzt werden, da sonst der IEC-Bus blockiert wird.

RADIAN:
stellt die trigonometrischen Routinen auf Bogenmaß ein.

RENUMBER z,s:
reorganisiert die Zeilennummern eines Basic-Programms. z ist dabei die Startzeile und s die Schrittweite. z und s dürfen nur Konstanten sein. Sollten dem Befehl keine Parameter folgen, so gilt z = 100 und s = 10.

RESTORE TO z:
positioniert den DATA-Pointer auf die Zeile z. Das Wort TO kann dabei wegfallen. RESTORE allein setzt den Pointer wie gewohnt auf den Programmstart.

RETURN TO z:
ermöglicht es, aus einem Unterprogramm in eine bestimmte Zeile z zurückzukehren.

RUN "name":
hat die gleiche Funktion wie DLOAD. Zusätzlich wird aber noch der String RUN +CHR$(13) in den Tastaturpuffer geschrieben. Das hat zur Folge, daß das gerade geladene Programm sofort gestartet wird.

PPRINT:
verhält sich wie PRINT. Die Zeichen werden aber, anstatt auf den Bildschirm, auf den Plotter (Nummer 6) ausgegeben.

QUIT:
führt ein Total-Reset aus. Mathematical Basic wird dabei gelöscht.

UNITL a: siehe DO.

Die Funktionen des Mathematical Basic

! a:
ist ein Äquivalent zur CHR$(a)-Funktion. Hierbei sind aber keine Klammern nötig. a ist nur als Konstante erlaubt.

&:
ist das »Hexadezimal-Vorzeichen« für Hex-Konstanten. Beispiel: &a02b ist das Äquivalent für 41003. Die Anzahl der Hex-Ziffern ist beliebig, es wird maximal eine 16-Bit-Zahl errechnet.

—:
entspricht dem Ausdruck CHR$(13) und kann auch genauso gehandhabt werden.

[a]:
hat die gleiche Wirkung wie die ABS-Funktion. Unterscheidet sich aber durch die bessere Selbstdokumentation.

ACS(a):
berechnet abhängig von der jeweiligen Einstellung durch RADIAN, DEGREE oder GRAD den Arcus-Cosinus von a. Die hier angesprochene Abhängigkeit gilt für alle trigonometrischen Funktionen.

ACT(a):
bestimmt den Arcus-Cotangens von a.

ASN(a):
ergibt den Arcus-Sinus von a.

CHR$(z,l):
ist eine Variante von CHR$(z). z entspricht dabei dem Zeichen des normalerweise nur 1 Byte langen Strings. Mit l läßt sich aber zusätzlich auch die Länge variieren. l ist dabei von 0 bis 255 definiert.

COT(a):
berechnet den Cotangens von a.

CRSCOL:
holt die momentane Spaltenposition des Cursors.

CRSLIN:
holt die momentane Zeilenposition des Cursors.

CVF(a$):
wandelt den fünf Zeichen langen String a$ in eine Fließkommazahl um.

CVI(a$):
wandelt den zwei Zeichen langen String a$ in eine Integerzahl um.

DEC(a$):
berechnet aus dem 4-Byte-String a$, der sich aus Hex-Ziffern aufbaut, eine dezimale 16-Bit-Zahl.

DIV(a,b,):
ist gleichwertig mit dem Ausdruck INT(a/b).

FRC(a):
holt die Nachkommazahl von a.

FUNCTION(a$):
wandelt den maximal 88 Zeichen langen String a$ in Interpretercode und berechnet ihn. Beispiel: y=FUNCTION("5+2"). y wird der Wert 7 zugewiesen. Diese Funktion darf nicht im Direktmodus stehen.

HEX$(a):
wandelt die 16-Bit-Zahl a in einen 4-Byte-Hex. String.

INSTR(a$,b$):
testet, an welcher Stelle a$ sich in b$ befindet. Ist a$ nicht in b$ enthalten, so ist das Ergebnis 0, ansonsten entspricht es der Position von a$ in b$. a$ und b$ müssen mindestens 1 Zeichen lang sein und a$ darf nicht länger als b$ sein. INSTR darf nicht im Direktmodus stehen.

LGD (a):
berechnet den dekadischen Logarithmus von a.

LGU(a,b):
bestimmt den Logarithmus von a zur Basis b.
MKF$(a):
wandelt die Zahl a in einen fünf Zeichen langen String.
MKI$(a):
wandelt die Integerzahl a in einen zwei Zeichen langen String.
MOD(a,b):
bestimmt den ganzzahligen Rest aus der Division von a durch b.
RANDOM:
entspricht dem Ausdruck RND(1).
RANDOM(a,b):
entspricht dem Ausdruck RND(1)*b+a.
TIME$:
ist eine modifizierte Form von TI$. Der Unterschied liegt darin, daß bei TIME$ noch zwei Trennzeichen eingefügt werden.
USRn:
entspricht im Prinzip der USR(a)-Funktion. Unterschiede liegen darin, daß n (was der Kennung A, B, C, D, E, F, G oder H entspricht) folgen muß, aber keine »Klammer-auf-Klammer-zu«-Sequenz. Diese Routine (Aufruf durch JSR$cef1) muß zusätzlich an USR-Routinen, die für die Standard-USR-Funktion ausgelegt sind, angehängt werden. Der Vorteil bei den USRn-Funktionen liegt nun darin, daß man den USR-String mit mehreren Parametern übergeben kann. Die dazu nötigen Routinen sind:

cefa	Test auf Klammer auf und nächstes Zeichen holen.
cd9e	Ausdruck holen und String- und Integer-Flag setzen.
cd8d	Test auf numerischen Ausdruck.
cd8f	Test auf String.
cef7	Test auf Klammer zu und nächstes Zeichen holen.
cefd	Test auf Komma und nächstes Zeichen holen.

Damit sind alle neuen Befehle und Funktionen vorgestellt. Bleibt nur noch, Ihnen viel Spaß beim Programmieren mit »Mathematical Basic« zu wünschen. (Wolfgang W. Wirth/ev)

```
100 rem---------------------------------
-----------------------------------
101 rem" * * *          M A T H E M A T I C
 A L   B A S I C   V1.02        * * * "
102 rem---------------------------------
-----------------------------------
103 rem" createt in october 1984 by W.Wi
rth,Th. Heuss Ring 20,6556 Woellstein "
104 rem---------------------------------
-----------------------------------
105 rem
106 rem
107 rem
108 clr:if peek(55)+peek(56)*256>29211 t
hen poke 55,27:poke56,114:clr
109 poke 36879,14:print chr$(147);chr$(1
3);chr$(14);chr$(5);spc(17);chr$(34)
110 dim cs(14)
111 rem
112 print"   Ladeprogramm fur"
113 print
114 print" ";chr$(18);" Math. Basic  V1.
02 "
115 print
116 print" Das Programm wird"
117 print
118 print" jetzt abgelegt."
119 print
120 print
121 rem
122 rem---------------------------- hilfs
routine ablegen -----------------------
123 rem
124 poke 0,76
125 poke 1,12
126 poke 2,2
127 rem
128 for i=512 to 543
129 :  read j
130 :  cs=cs+j
131 :  poke i,j
132 next
133 rem
134 if cs=2635 then 148
135 rem
136 print
137 print
138 print" Datafehler im"
139 print
140 print" Hilfsrout.-Block !":end
141 rem
142 rem-------------------------- hilfs
routinenblock -------------------------
143 rem
144 data136,177,34,201,58,144,2,233,8,23
3
145 data47,96,32,130,215,32,,2,133,102,3
2,
146 data2,10,10,10,10,5,102,76,148,215
147 rem
148 rem-------------------------- haupt
programm ablegen ----------------------
149 rem
150 for i=0 to 3556
151 :  print i;chr$(145)
152 :  read j$:l=usr(j$)
153 :  cs(i/240)=cs(i/240)+l:poke 29211+
i,l
154 next
155 rem
156 rem-------------------------- fehle
rerkennung ----------------------------
157 rem
158 for i=0 to 14
159 :  read cs:if cs(i)=cs then 163
160 :  print" Fehler im Block von"
161 :  print" Zeile";171+i*10;"bis";180+
i*10:print
162 :  print:ef=1
163 next
164 rem
165 if ef then end
166 sys 30414
167 rem
168 rem-------------------------- mathe
matical basic v1.02 -------------------
169 rem
170 data48,d2,48,d2,48,d2,48,d2,48,d2,48
,d2,48,d2,48,d2,20,f1,ce,20,82,d7,c9,04
171 datad0,62,88,b1,22,a2,0f,dd,73,73,f0
,05,ca,10,f8,30,53,8a,ea,ea,ea,ea,a2,04
172 data6a,66,62,66,63,ca,d0,f8,88,10,e0
,a2,90,38,4c,49,dc,20,f1,ce,a5,14,48,a5
173 data15,48,20,f7,d7,a9,04,20,7d,d4,a8
,a5,14,20,81,72,a5,15,20,81,72,68,85,15
174 data68,85,14,4c,fb,d6,48,29,0f,20,8c
,72,68,4a,4a,4a,4a,aa,bd,73,73,88,91,62
175 data60,4c,58,d6,4c,48,d2,20,a6,d3,20
,fa,ce,20,9e,cd,20,a3,d6,c9,00,f0,ed,48
```

Listing »Mathematical Basic«

```
176 data8a,48,98,48,20,fd,ce,20,9e,cd,20
,a3,d6,85,60,aa,f0,da,20,f7,ce,68,85,65
177 data68,85,64,68,85,66,a5,60,c5,66,90
,c5,a9,00,85,0d,a2,01,a0,ff,c8,c4,66,d0
178 data06,8a,4c,94,d7,a0,00,b1,22,d1,64
,f0,ef,e6,22,d0,02,e6,23,e8,c6,60,a5,60
179 datac5,66,b0,e9,4c,f7,d8,20,73,00,20
,8a,cd,4c,f7,d7,4c,08,cf,20,f1,ce,6c,01
180 data00,c9,28,f0,f6,38,e9,41,c9,08,b0
,ec,0a,69,1b,8d,21,73,20,73,00,6c,00,72
181 data4c,b3,d3,c9,b7,d0,f9,20,73,00,c9
,a4,d0,08,20,fa,72,84,01,85,02,60,38,e9
182 data41,c9,08,b0,c3,0a,48,20,73,00,c9
,a4,d0,ba,20,fa,72,68,aa,a5,14,9d,1b,72
183 dataa5,15,9d,1c,72,60,08,c9,22,f0,03
,4c,20,77,a2,04,78,86,c6,bd,f8,ed,9d,76
184 data02,ca,d0,f7,58,4c,58,7a,30,31,32
,33,34,35,36,37,38,39,41,42,43,44,45,46
185 dataa9,00,85,62,85,63,20,73,00,a0,0f
,d9,73,73,f0,06,88,10,f8,4c,56,72,98,2a
186 data2a,2a,2a,a2,04,2a,26,63,26,62,ca
,d0,f8,f0,df,00,00,ee,0a,0a,ee,0a,0a,ee
187 data0a,0a,ee,0a,0a,ee,0a,0a,ee,0a,0a
,ee,0a,0a,00,c9,26,f0,bd,c9,5b,d0,06,20
188 data73,00,4c,57,7e,c9,21,d0,08,48,48
,20,73,00,4c,5c,7d,c9,5f,f0,03,4c,92,ce
189 data48,48,20,73,00,4c,75,7d,20,fa,ce
,20,9e,d7,a8,8a,48,c0,2c,d0,05,20,f1,d7
190 data8a,ac,a9,01,20,7d,d4,20,f7,ce,68
,a0,00,f0,03,91,62,c8,c4,61,90,f9,4c,fb
191 datad6,a2,06,ac,a2,04,a0,60,20,50,fe
,20,14,f3,20,9d,ca,ea,ea,4c,b7,cb,a2,04
192 databd,4e,76,9d,a0,02,ca,d0,f7,60,20
,79,00,f0,13,a0,00,20,4e,74,f0,0c,c9,2c
193 datad0,6f,20,73,00,20,4e,74,d0,67,60
,b0,64,20,6b,c9,a5,14,99,a1,02,c8,ea,ea
194 dataea,a5,15,99,a1,02,c8,4c,79,00,20
,79,00,f0,4a,08,a0,00,20,b7,74,48,20,13
195 datac6,68,28,f0,07,c9,ab,d0,38,20,73
,00,08,a0,02,20,b7,74,d0,2d,28,d0,04,c6
196 data15,c6,af,a5,5f,85,7a,a5,60,85,7b
,20,13,c6,a0,01,90,0a,b1,5f,aa,88,b1,5f
197 data85,5f,86,60,a5,ae,c5,ac,a5,af,e5
,ad,ea,ea,ea,b0,99,4c,08,cf,20,6b,c9,a6
198 data14,96,ac,a6,15,96,ad,4c,79,00,20
,29,74,20,35,74,20,b3,75,20,c0,75,f0,0d
199 data20,0a,76,b0,05,20,1a,76,f0,f1,4c
,48,d2,20,b3,75,20,c0,75,d0,2c,20,b3,75
200 data20,c0,75,f0,12,a0,02,b9,ab,00,91
,7a,88,d0,f8,20,0a,76,20,1a,76,f0,e9,20
201 datac4,75,a5,7a,85,2d,a5,7b,20,57,c6
,20,33,c5,4c,74,c4,20,c0,75,a9,80,85,0f
202 dataa9,80,2c,a9,40,45,0f,85,0f,20,73
,00,a8,f0,b9,c9,22,f0,ed,a6,0f,d0,f2,c9
203 data8f,f0,e8,a2,08,dd,55,76,f0,05,ca
,d0,f8,f0,e2,20,cf,75,20,73,00,a2,03,dd
204 data52,76,f0,f3,ca,d0,f8,20,79,00,b0
,d0,20,6b,c9,20,b3,75,20,c0,75,f0,13,20
205 datac0,75,c5,15,d0,04,c4,14,f0,0e,20
,0a,76,20,1d,76,f0,e8,a0,f9,a9,ff,d0,04
206 dataa4,ad,a5,ac,20,23,76,20,fe,75,86
,ae,20,73,00,e6,ae,a1,ae,f0,17,90,05,a0
207 dataff,20,d9,75,a1,ae,81,7a,20,c4,75
,c9,3a,b0,e8,20,98,e3,10,e3,20,79,00,b0
208 data9c,a0,01,20,d9,75,f0,f4,a2,02,bd
,a0,02,95,ab,ca,d0,f8,4c,8e,c6,20,c4,75
209 dataa8,e6,7a,d0,02,e6,7b,a2,00,a1,7a
,60,a2,02,b5,79,95,59,ca,d0,f9,60,20,cf
210 data75,a9,03,85,c2,b1,7a,d0,02,e6,c2
,c8,85,c1,b1,7a,48,a5,c1,81,7a,f0,04,a9
211 data04,85,c2,20,c4,75,68,c6,c2,d0,e9
,a2,02,b5,59,95,79,86,af,ca,d0,f7,60,a2
212 datafe,18,b5,ae,7d,a5,01,95,ae,e8,d0
,f6,c9,fa,60,20,c0,75,20,c4,75,d0,fb,60
213 data84,62,85,63,a2,90,38,20,49,dc,4c
,dd,dd,20,65,74,a0,00,b1,5f,91,7a,e6,5f
214 datad0,02,e6,60,20,c4,75,a5,5f,c5,2d
,a5,60,e5,2e,90,e9,4c,05,75,64,00,0a,00
215 dataab,a4,2c,9b,8a,a7,89,8d,cb,8c,8e
,00,00,00,00,00,00,a5,90,f0,05,68,68,4c
216 datade,f6,20,e1,ff,f0,f6,20,f9,f1,c9
,20,d0,05,20,f9,f1,f0,fb,4c,19,ef,a0,00
217 dataa2,08,84,90,20,52,fe,a9,01,a2,60
,a0,c3,20,49,fe,20,9d,f4,a9,08,20,b4,ff
218 dataa9,00,20,96,ff,20,64,76,20,64,76
,20,64,76,20,64,76,20,64,76,85,63,20,64
219 data76,85,62,20,d1,dd,20,64,76,aa,f0
,06,20,42,e7,4c,b9,76,20,d7,ca,20,d7,ca
220 data4c,a6,76,a2,ff,78,9a,d8,20,8d,fd
,a2,1b,a0,72,cc,84,02,90,07,d0,08,ec,83
221 data02,b0,03,20,7b,fe,20,a9,7e,20,a4
,e3,a5,2b,a4,2c,20,08,c4,a9,98,a0,7d,20
222 data0f,e4,4c,81,e3,a2,08,a0,0f,20,52
,fe,a9,01,a2,72,a0,c3,20,49,fe,4c,9d,f4
223 data20,26,77,4c,60,c6,20,26,77,4c,e9
,[illegible],00,20,26,77,4c,72,c8,48,a9,00,8d,5e
224 data77,8d,69,79,68,60,ad,5e,77,f0,4a
,20,9e,cd,a5,61,f0,04,ce,5e,77,60,ae,5e
225 data77,bd,98,77,85,39,bd,a0,77,85,3a
,bd,a8,77,85,7a,bd,b0,77,85,7b,d0,20,4c
226 data35,c4,a2,00,e0,08,b0,f7,a5,39,9d
,99,77,a5,3a,9d,a1,77,a5,7a,9d,a9,77,a5
227 data7b,9d,b1,77,ee,5e,77,20,79,00,4c
,fd,7e,a9,89,85,22,a9,77,4c,45,c4,55,4e
228 data54,49,4c,20,57,49,54,48,4f,55,54
,20,44,cf,bf,ff,ff,bf,ff,ff,ff,ff,ff,ff
229 dataff,ff,ff,ff,ff,ff,ff,ff,ff,ff,ff
,ff,ff,ff,ff,ff,ff,ff,ff,ff,ff,ff,4c,e3
230 datac8,4c,1d,c8,f0,fb,20,fa,77,20,79
,00,20,6b,c9,20,13,c6,90,ea,a5,5f,a4,60
231 datae9,01,4c,24,c8,4c,eb,c8,4c,e0,c8
,a9,ff,85,4a,20,8a,c3,9a,c9,8d,d0,f1,20
232 data79,00,f0,e9,28,28,28,28,28,20,fa
,77,4c,a0,c8,c9,a4,d0,4b,4c,73,00,20,fa
233 datace,20,8a,cd,20,ea,d9,a2,24,a0,78
,20,d4,db,20,fd,ce,20,8a,cd,20,ea,d9,a9
234 data24,a0,78,20,0f,db,4c,f7,ce,81,b1
,72,17,f8,7f,5e,5b,d8,a9,20,f1,ce,20,ea
235 datad9,a9,29,a0,78,4c,28,da,4c,bb,e0
,c9,28,d0,f9,20,73,00,20,8a,cd,a2,60,a0
236 data7c,20,d4,db,20,fd,ce,20,8a,cd,a2
,70,a0,78,20,d4,db,20,f7,ce,20,bb,e0,a9
237 data70,a0,78,20,28,da,a9,60,a0,7c,4c
,67,d8,00,00,00,00,00,20,9e,d7,e0,08,b0
238 data1f,8e,86,02,20,f1,d7,e0,10,b0,15
,8a,38,2a,0a,0a,0a,48,20,f1,d7,e0,08,b0
239 data07,68,05,65,8d,0f,90,60,4c,48,d2
,f0,0b,20,9e,d7,e0,7f,b0,f4,8a,09,80,ae
240 dataa9,fa,48,20,79,00,f0,04,20,f1,d7
,ad,a2,05,68,e0,01,90,22,86,fe,8d,0c,90
```

```
241 dataa9,0f,a8,0d,0e,90,8d,0e,90,a6,fe
,66,61,c6,60,d0,fa,ca,d0,f7,ce,0e,90,88
242 datad0,ef,8c,0c,90,60,ad,16,7f,4a,29
,03,a8,88,8c,69,79,60,a2,5b,a0,7c,20,d4
243 datadb,46,66,a9,bc,a0,d9,20,5b,dc,d0
,0e,a9,dd,a0,e2,20,a2,db,a9,5b,a0,7c,4c
244 data28,da,a9,5b,a0,7c,20,a2,db,20,b4
,df,a9,5b,a0,7c,20,28,da,20,49,d8,20,49
245 datad8,20,71,df,a9,5b,a0,7c,20,0f,db
,4c,0b,e3,20,ed,78,a9,dd,a0,e2,4c,50,d8
246 data20,0b,e3,4c,34,79,a9,dd,a0,e2,20
,50,d8,4c,b1,e2,7b,0e,fa,35,12,7b,00,ad
247 datafc,90,86,65,2e,e0,d4,86,7e,a5,dd
,5e,4b,50,55,5a,20,f1,ce,a9,5f,a0,00,f0
248 data0c,88,8d,71,79,b9,5f,79,a0,79,4c
,28,da,60,20,63,79,4c,68,e2,20,63,79,4c
249 data61,e2,20,63,79,4c,b1,e2,20,63,79
,4c,41,79,20,f1,ce,20,ed,78,a9,61,4c,68
250 data79,20,f1,ce,20,31,79,4c,97,79,20
,f1,ce,20,0b,e3,4c,97,79,20,f1,ce,20,3b
251 data79,4c,97,79,20,f1,ce,20,58,dc,a2
,2e,a0,7a,20,d4,db,20,cc,dc,a9,2e,a0,7a
252 data4c,50,d8,20,fa,ce,20,8a,cd,20,fd
,ce,a2,33,a0,7a,20,d4,db,20,8a,cd,a2,38
253 dataa0,7a,20,d4,db,a9,bc,a0,d9,20,0f
,db,a9,33,a0,7a,20,28,da,20,ba,79,a9,38
254 dataa0,7a,20,28,da,8f,00,00,79,b3,dc
,4c,f7,ce,20,fa,ce,20,8a,cd,a2,29,a0,7a
255 data20,d4,db,20,fd,ce,20,8a,cd,a9,29
,a0,7a,20,0f,db,20,cc,dc,4c,f7,ce,00,00
256 data00,00,00,00,00,00,00,00,00,00,00
,00,00,00,00,00,00,00,4c,54,e2,a2,08,a0
257 data00,20,50,fe,20,79,00,d0,f1,a9,01
,a2,38,a0,e4,4c,49,fe,a9,01,2c,a9,00,85
258 data0a,20,40,7a,4c,6c,e1,4c,08,cf,4c
,48,d2,4c,58,d6,4c,ab,d3,20,f1,ce,38,b0
259 data04,20,9e,cd,18,08,a6,3a,e8,f0,ed
,20,82,d7,f0,e2,c9,59,b0,e1,8a,f0,02,b1
260 data22,99,00,02,88,10,f8,28,a5,7a,48
,a5,7b,48,08,a9,00,a0,02,85,7a,84,7b,20
261 data79,c5,28,b0,12,20,73,00,20,fd,7e
,20,79,00,c9,3a,f0,f3,a8,f0,08,d0,a8,20
262 data73,00,20,8a,cd,68,85,7b,68,85,7a
,60,20,f1,ce,20,8d,cd,20,bf,d1,a5,64,48
263 dataa5,65,48,a9,02,20,7d,d4,68,a0,01
,91,62,4c,f6,d6,20,f1,ce,20,8d,cd,a2,ff
264 data20,cc,db,a9,05,20,7d,d4,a0,04,b9
,ff,00,91,62,88,10,f8,4c,fb,d6,41,53,ce
265 data41,43,d3,43,4f,d4,41,43,d4,4c,47
,c4,4c,47,d5,43,56,c9,43,56,c6,4d,4b,49
266 dataa4,4d,4b,46,a4,43,52,53,4c,49,ce
,43,52,53,43,4f,cc,54,49,4d,45,a4,49,4e
267 data53,54,d2,48,45,58,a4,44,45,c3,46
,55,4e,43,54,49,4f,ce,4d,4f,c4,44,49,d6
268 data46,52,c3,52,41,4e,44,4f,cd,52,41
,44,49,41,ce,44,45,47,52,45,c5,47,52,41
269 datac4,51,55,49,d4,a7,4c,4f,43,41,54
,c5,42,45,45,d0,43,4f,4c,4f,d2,43,4c,d3
270 data45,58,45,43,55,54,c5,44,4c,4f,41
,c4,44,53,41,56,c5,44,56,45,52,49,46,d9
271 data43,41,54,41,4c,4f,c7,44,49,52,45
,43,54,4f,52,d9,49,4e,49,54,49,41,4c,49
272 data5a,c5,44,cf,55,4e,54,49,cc,50,50
,52,49,4e,d4,4c,50,52,49,4e,d4,52,45,4e
273 data55,4d,42,45,d2,44,45,4c,45,54,c5
,00,00,00,00,00,00,00,00,00,00,00,00,00
274 data00,00,00,00,00,00,00,00,00,00,00
,00,00,00,00,00,00,00,00,00,00,00,00,00
275 data00,00,00,00,00,00,00,00,00,00,00
,00,00,31,c8,42,c7,1e,cd,f8,c8,a5,cb,bf
276 datacb,81,d0,06,cc,a5,c9,a0,c8,59,73
,91,7e,bf,77,83,c8,de,77,3b,c9,2f,c8,4b
277 datac9,2d,d8,65,e1,53,e1,62,e1,26,73
,24,d8,80,ca,a0,ca,57,c8,9c,c6,13,77,86
278 dataca,27,e1,bb,e1,c4,e1,7b,cb,19,77
,a4,d3,ad,a4,d6,c8,4c,a2,d3,20,a6,db,84
279 data0d,60,20,40,7a,4c,56,e1,00,00,00
,00,00,08,cf,08,cf,00,00,00,00,4d,7e,4d
280 data7e,4d,7e,0c,73,4d,7e,4d,7e,4d,7e
,4d,7e,4d,7e,4d,7e,7f,79,79,79,85,79,a5
281 data79,4d,7e,4d,7e,4d,7e,4d,7e,4d,7e
,eb,73,4d,7e,4d,7e,4d,7e,08,cf,91,79,9c
282 data79,8b,79,ae,79,2e,78,01,78,7a,7d
,8b,7d,c7,7a,e3,7a,49,7c,46,7c,00,7d,9a
283 data72,5c,72,2b,72,6e,7a,ce,79,09,7a
,b7,79,3e,78,e1,78,e1,78,e1,78,22,fd,f8
284 datac8,3c,7d,9e,78,75,78,5f,e5,74,7a
,58,7a,55,7c,55,7a,81,76,81,76,00,77,5d
285 data77,31,77,14,74,17,74,c5,74,30,76
,08,cf,08,cf,08,cf,08,cf,08,cf,08,cf,08
286 datacf,08,cf,08,cf,20,84,cf,84,5e,88
,84,71,a0,06,84,5d,a0,24,20,68,de,a9,00
287 data8d,07,01,ad,04,01,8d,06,01,ad,03
,01,8d,05,01,ad,02,01,8d,03,01,ad,01,01
288 data8d,02,01,a9,3a,8d,01,01,8d,04,01
,4c,[illegible]d,d4,4c,48,d2,c9,2c,f0,0f,20,9e,d7
289 dataca,e0,17,b0,f1,86,d6,20,79,00,f0
,0a,20,f1,d7,ca,e0,16,b0,e2,86,d3,4c,87
290 datae5,a5,14,48,a5,15,48,20,79,00,20
,6b,c9,a5,15,d0,cd,a6,14,68,85,15,68,85
291 data14,ad,a2,0d,4c,ef,d6,20,f1,ce,20
,a3,d6,c9,02,d0,b5,86,64,84,65,4c,61,cf
292 data20,f1,ce,20,a3,d6,c9,05,d0,a4,4c
,4f,7c,93,0e,05,b0,c0,c0,c0,c0,c0,c0,c0
293 datac0,c0,c0,c0,c0,c0,c0,c0,c0,c0,c0
,c0,c0,ae,dd,9c,20,20,4f,50,45,52,41,54
294 data49,4e,47,20,53,59,53,54,45,4d,20
,20,05,dd,dd,20,cd,c1,d4,c8,2e,20,c2,c1
295 datad3,c9,c3,20,20,d6,31,2e,30,32,20
,dd,dd,9c,42,59,20,d7,2e,d7,49,52,54,48
296 data20,30,36,37,30,33,2f,38,36,38,05
,dd,ad,c0,c0,c0,c0,c0,c0,c0,c0,c0,c0,c0
297 datac0,c0,c0,c0,c0,c0,c0,c0,c0,bd,0d
,0d,20,42,41,53,49,43,20,4d,45,4d,4f,52
298 data59,3a,11,0d,20,00,3a,c4,83,c4,1b
,7f,d6,7f,f4,7e,d4,7e,bf,ea,85,7e,68,7e
299 data0a,f4,4a,f3,c7,f2,09,f3,f3,f3,0e
,f2,7a,f2,70,f7,f5,f1,ef,f3,d2,fe,49,f5
300 data85,f6,68,68,ad,16,7f,48,aa,4c,ad
,cf,20,8a,cd,46,66,a9,5d,4c,ff,ce,00,4c
301 data56,ff,4c,de,fe,48,8a,48,98,48,ad
,1d,91,10,f0,2d,1e,91,aa,29,02,f0,eb,2c
302 data11,91,20,34,f7,20,e1,ff,d0,dd,20
,a9,7e,6c,02,c0,4c,3b,c9,4c,7a,77,20,9e
303 datacd,a5,61,f0,f3,20,79,00,c9,a7,d0
```

Listing »Mathematical Basic« (Fortsetzung)

```
,ef,20,73,00,b0,ea,4c,a0,c8,00,00,78,a2
304 data2d,a0,7e,20,56,fd,a2,0b,bd,21,7e
,9d,00,03,ca,10,f7,20,f9,fd,20,18,e5,a9
305 data0e,8d,0f,90,a9,01,8d,86,02,ea,60
,4c,f3,dc,4c,c2,73,a9,00,85,0d,20,73,00
306 data90,f1,c9,e1,b0,f0,c9,b4,90,ec,20
,0e,7f,4c,8d,cd,4c,08,cf,4c,a5,c9,4c,12
307 datac8,20,73,00,20,fd,7e,4c,ae,c7,f0
,ce,aa,10,ec,c9,cb,f0,eb,c9,a3,90,04,c9
308 datae1,90,dd,0a,8d,16,7f,20,73,00,6c
,c8,7c,00,00,00,a6,7a,a0,04,84,0f,bd,00
309 data02,10,07,c9,ff,f0,3e,e8,d0,f4,c9
,20,f0,37,85,08,c9,22,f0,56,24,0f,70,2d
310 datac9,3f,d0,04,a9,99,d0,25,c9,30,90
,04,c9,3c,90,1d,84,71,a0,00,84,0b,88,86
311 data7a,ca,c8,e8,bd,00,02,38,f9,9e,c0
,f0,f5,c9,80,d0,30,05,0b,a4,71,e8,c8,99
312 datafb,01,b9,fb,01,f0,59,38,e9,3a,f0
,04,c9,49,d0,02,85,0f,38,e9,55,d0,9f,85
313 data08,bd,00,02,f0,df,c5,08,f0,db,c8
,99,fb,01,e8,d0,f0,a6,7a,e6,0b,c8,b9,9d
314 datac0,10,fa,b9,9e,c0,d0,b4,a0,ff,ca
,c8,e8,bd,00,02,38,f9,00,7b,f0,f5,c9,80
315 datad0,02,f0,ad,a6,7a,e6,0b,c8,b9,ff
,7a,10,fa,b9,00,7b,d0,e2,bd,00,02,10,9b
316 data4c,09,c6,4c,ef,c6,b1,5f,4c,1a,c7
,c9,cc,90,f7,c9,ff,f0,f3,24,0f,30,ef,84
317 data49,38,e9,cb,aa,a0,ff,ca,f0,08,c8
,b9,00,7b,10,fa,30,f5,c8,b9,00,7b,30,d3
318 data20,47,cb,d0,f5
319 rem
320 rem------------------------- pruef
summen ----------------------------
321 rem
322 data 31231
323 data 25675
324 data 28397
325 data 31427
326 data 30997
327 data 33043
328 data 30573
329 data 30423
330 data 28240
331 data 26807
332 data 21966
333 data 26340
334 data 31124
335 data 29634
336 data 27953

ready.
```

Listing »Mathematical Basic« (Schluß)

Ohne gutes Werkzeug geht es nicht:

SMON Teil 2

Der Maschinensprache-Monitor SMON wird immer leistungsfähiger. Dieser 2. Teil erweitert ihn um wichtige Ausgabe-Routinen, läßt das Verschieben eines Programms mit und ohne Adreßumrechnung zu und kann Zahlen vom Dezimal- in das Binärsystem und umgekehrt umrechnen.

Wir hoffen, daß wir Ihnen in der letzten Ausgabe nicht zuviel zugemutet haben, und daß sich Ihre wunden Finger inzwischen wieder erholen konnten. Bestimmt haben Sie im vergangenen Monat schon eifrig mit dem neuen Monitor gearbeitet und sind inzwischen mit den bisherigen Befehlen vertraut. Denn nun folgt der zweite Teil und mit diesem natürlich wieder einige neue Befehle, die es zu erklären gilt.

Und das bieten wir Ihnen heute:

I/O-SET, LOAD, SAVE, PRINTER-SET, die verschiedenen Zahlenumrechnungen (HEX-DEZ-BIN-ADD-SUB), OCCUPY, CONVERT, VERSCHIEBEN und WRITE.

I/O-SET

I 01 legt die Device-Nummer für LOAD und SAVE auf 1 (Kassette). Jedes Laden und Abspeichern erfolgt jetzt auf das angegebene Gerät. Die voreingestellte Device-Nummer ist 8 (für die Floppy also: I 08). Wenn Sie nur mit der Floppy arbeiten, brauchen Sie diesen Befehl also nicht.

LOAD

L "Name" lädt ein Programm vom angegebenen Gerät (wie oben beschrieben) an die Originaladresse in den Speicher. Die Basic-Zeiger bleiben bei diesem Ladevorgang unbeeinflußt, das heißt, sie werden nicht verändert.

Beispiel: Unser Monitor soll an seiner Originaladresse ($C000) im Speicher stehen. Also brauchen Sie ihn nur mit »L "SMON"« zu laden, damit er dort erscheint. Wenn Sie einmal ein Programm an eine andere als die Originaladresse laden wollen, dann bietet Ihnen SMON dazu folgende Möglichkeit: 'L "Name" ADRESSE lädt ein Programm an die angegebene Adresse. Nehmen Sie doch bitte noch einmal unser letztes Test-Programm und geben es mit dem Assembler ab Adresse $4000 ein. Speichern Sie es mit »S "SUPERTEST" 4000 4023« ab und laden es dann

1. an die Originaladresse (L "SUPERTEST") und

2. an eine andere Adresse (mit L "SUPERTEST"5000 zum Beispiel nach $5000).

Schauen Sie sich danach mit dem Disassembler-Befehl beide Routinen einmal an. Sie werden feststellen, daß beide Programme zwar bis auf die BRANCH-Befehle gleich aussehen, daß das Programm in $5000 aber nicht funktionieren kann, da es eine falsche Adresse verwendet (5002 LDA 400E,Y). Ein anderes Beispiel dazu: Ein Autostart-Programm beginnt bei $0120, läßt sich aber in diesem Bereich nicht untersuchen, da dort der Prozessor-STACK (im Bereich von $0100 bis $01FF) liegt, der vom Prozessor selbst ständig verändert wird. Wenn Sie nun L "Name" 4120 eingeben, befindet sich das Programm anschließend bei $4120 (nicht an der Originaladresse $0120) und Sie können es ohne Einschränkungen — von den falschen Absolut-Adressen abgesehen — disassemblieren.

SAVE

S "Name", ANFADR ENDADR speichert ein Programm von ANFADR bis ENDADR-1 unter »Name« auf die Floppy ab, da diese — wie wir ja inzwischen wissen — das voreingestellte Gerät ist. Wenn Sie auf Kassette abspeichern wollen, setzen Sie vorher mit »I 01« die Device-Nummer auf 1.

Beispiel: S"SUPERTEST"4000 4020 speichert das Programm mit dem Namen »SUPERTEST« (es steht im Speicher von $4000 bis $401F) auf Diskette ab. Bitte beachten Sie auch bei diesem Befehl, daß die Endadresse auf das nächste Byte hinter dem Programm gesetzt wird.

PRINTER-SET

P 02 setzt die Primäradresse für den Drucker auf 2. Voreingestellt ist hier die 4 als Gerätenummer (zum Beispiel für Commodore-Drucker). Vielleicht haben Sie es ja schon bemerkt: Bei allen Ausgabe-Befehlen (wie D, M etc.) können Sie auch den Drucker ansprechen, wenn Sie das Kommando geshiftet eingeben. Die Ausgabe erfolgt dann gleichzeitig auf Bildschirm und Drucker. (Beachten Sie bitte die Änderung für die Druckerausgabe am Schluß des Artikels.)

Ein bißchen Rechnerei

Die folgende Befehlsgruppe enthält Befehle zur Zahlenumrechnung. Sie wissen ja: Der Mensch mit seinen zehn Fingern neigt eher zur dezimalen Rechenweise, aber der Computer bevorzugt das Binärsystem, weil er nur zwei Finger hat (siehe Netzstecker). Ein Kompromiß ist das Hexadezimalsystem, denn das versteht keiner von beiden. Um Verständnis-Schwierigkeiten mit Ihrem Liebling aus dem Weg zu gehen, haben Sie aber SMON.

UMRECHNUNG DEZ→HEX

(Dezimalzahl) rechnet die Dezimalzahl in die entsprechende Hexadezimalzahl um. Hierbei können Sie die Eingabe in beliebiger Weise vornehmen, da SMON Zahlen bis 65535 umrechnet. Beispiel: #12, #144, #3456, #65533 und so weiter.

UMRECHNUNG HEX→DEZ

$ (Hexadezimalzahl) rechnet die Hexadezimalzahl in die entsprechende Dezimalzahl um. Die Eingabe muß hierbei zweistellig beziehungsweise vierstellig erfolgen. Ist diese Zahl kleiner als $100 (=255), wird zusätzlich auch der Binärwert ausgegeben.

Beispiel: $12, $0012, $0D, $FFD2, etc. In den ersten drei Beispielen erfolgt die Anzeige auch in binärer Form.

UMRECHNUNG BINÄR→HEX,DEZ

% (Binärzahl (achtstellig) rechnet die Binärzahl in die entsprechenden Hexa- und Dezimalzahlen um. Bei diesem Befehl müssen Sie genau acht Binärzahlen eingeben. Falls Sie einmal versehentlich mehr eingeben sollten, werden nur die ersten acht zur Umrechnung herangezogen. Beispiel: %00011111, %10101011

ADD-SUB

? 2340+156D berechnet die Summe der beiden vier (!)-stelligen Hex-Zahlen. Neben der Addition ist auch Subtraktion möglich.

Programme auf dem Rangierbahnhof

OCCUPY (Besetzen)

O (ANFADR ENDADR HEX-Wert) belegt den angebenen Bereich mit dem vorgegebenen HEX-Wert. Beispiel:O 5000 8000 00 füllt den Bereich von $5000 bis $7FFF mit Nullen.

Man kann mit »OCCUPY« aber nicht nur Speicherbereiche löschen, sondern auch mit beliebigen Werten belegen. Häufig hat man das Problem, festzustellen, welcher Speicherplatz von einem Programm wirklich benutzt wird. Wir füllen den in Frage kommenden Bereich dann zuerst zum Beispiel mit »AA« und laden dann unser Programm. Probieren Sie bitte das folgende Beispiel: Füllen Sie den Speicherbereich von $3000 bis $6000 mit $AA, und laden Sie dann unser SUPERTEST-Programm. Beim Disassemblieren können Sie erkennen, daß unser kleines Programm exakt zwischen vielen AAs eingebettet ist.

WRITE

W (ANFADRalt ENDADRalt ANFADRneu) verschiebt den Speicherbereich von ANFADRalt bis ENDADRalt nach ANFADRneu ohne Umrechnung der Adressen! Unser kleines Testprogramm möge noch einmal als Beispiel dienen: W 4000 4020 6000 verschiebt das oben angesprochene Programm von $4000 nach $6000.

Hierbei werden weder die absoluten Adressen umgerechnet noch die Tabellen geändert. Letzteres ist sicherlich erwünscht, aber denken Sie daran, daß das verschobene Programm nun nicht mehr lauffähig ist, da die absoluten Adressen nicht mehr stimmen (zum Beispiel bei dem Befehl LDA 400E,Y). Falls Sie jetzt »G6000« eingeben, um das Programm zu starten, werden Sie sich sicherlich wundern, daß es dennoch läuft. Doch löschen Sie einmal das Programm in $4000 (mit »O4000 4100 AA«) und starten das Programm in $6000 noch einmal! Seltsam, nicht? Abhilfe schafft der nächste Befehl.

VARIATION

V (ANFADRalt ENDADRalt ANFADRneu ANFADR ENDADR) rechnet alle absoluten Adressen im Bereich von ANFADR bis ENDADR, die sich auf ANFADRalt bis ENDADRalt beziehen, auf ANFADRneu um. Kompliziert? Nicht, wenn Sie sich klarmachen, daß die ersten drei Adressen exakt den Eingaben beim »W«-Befehl entsprechen. Neu hinzukommen nur die beiden Adressen für den Bereich, in dem die Änderung tatsächlich erfolgt.

Um unser mit »W« schon verschobenes Programm auch wieder lauffähig zu machen, geben Sie folgendes ein: V 4000

4020 6000 6000 600E. Damit werden alle Absolutadressen, die im Bereich von $6000 bis $600E — dahinter steht die Tabelle — liegen und sich bisher auf $4000 bis $4020 bezogen haben, auf den neuen Bereich umgerechnet. Probieren geht wie immer über kapieren.

Eine Zusammenfassung dieser beiden Befehle ermöglicht:

CONVERTIEREN
verschieben eines Programmes mit Adreßumrechnung)
C (ANFADRalt ENDADRalt ANFADRneu ANFADRges ENDADRges) verschiebt das Programm von ANFADRalt bis ENDADRalt zur ANFADRneu und zwar mit Umrechnung der Adressen zwischen ANFADRges und ENDADRges

An unserem kleinen Testprogramm läßt sich wieder einmal demonstrieren, wie der Befehl eingesetzt wird. Laden Sie es also mit »L"SUPERTEST"« und schauen es mit »D 4000« an. Jetzt wollen wir an der Adresse $4008 einen 3-Byte-Befehl einfügen: C 4008 4020 400B 4000 4011 verschiebt das Programm von $4008 bis $4020 zur neuen Anfangsadresse $400B. Dabei werden im Bereich von $4000 bis $4011 (neue Endadresse des »aktiven« Programmes!) die Sprungadressen umgerechnet. Nun können Sie ab Adresse $4008 einen 3-Byte-Befehl einfügen, zum Beispiel STY 0286. Dazu geben Sie bitte ein:

```
A 4008
4008 STY 0286
F
```

Überzeugen Sie sich davon, daß SMON die Befehle korrekt umgerechnet hat, indem Sie unser Beispiel disassemblieren (D 4000) und anschließend mit G 4000 starten. Besitzer eines Farbmonitors werden in helle Begeisterung ausbrechen. Vorsicht ist geboten, wenn Tabellen oder Text vorhanden sind. SMON wird versuchen, diese als Befehle zu disassemblieren und gegebenenfalls umzurechnen. Dabei können unvorhersehbare Verfälschungen auftreten. Aus diesem Grunde ist im Beispiel die Endadresse des zu ändernden Bereiches auf $4011 und nicht etwa auf $4023 gelegt worden. Wenn Sie größere Programme zu verschieben haben, sollten Sie die Kommandos W und V anwenden, beziehungsweise einen Assembler einsetzen, der es Ihnen gestattet, beliebige Einfügungen, Verschiebungen und sonstige Änderungen vorzunehmen. Das C-Kommando eignet sich in erster Linie für kleinere Änderungen innerhalb eines Programms.

Der Blick hinter die Kulissen

Wie beim letzten Mal wollen wir noch einen kleinen Blick auf das Programm werfen. Wir haben zwei häufig vorkommende Programmteile ausgewählt. Wenn Sie nach erfolgreichem Eintippen der neuen DATAs einmal mit »D C84F« die »LOADSAVE«-Routine listen, sehen Sie, daß diese sehr wenig Platz beansprucht. Das Geheimnis dieser Beschränkung liegt in der Tatsache begründet, daß wir hier auf Betriebssystem-Routinen zurückgegriffen haben. Doch dazu nachher mehr; erst einmal die angesprochene Routine von Anfang an:

```
LOADSAVE   LDY   #$02
           STY   *$BC
           DEY
           STY   *$B9
           STY   *$BB
           DEY
           STY   *$B7
```

Die Speicherstellen $BB/$BC enthalten jetzt die Adresse $0201, also den Beginn des Basic-Eingabepuffers. In $B9 befindet sich der Wert 01, das bedeutet, daß die Sekundäradresse für absolutes Laden voreingestellt ist. Die Speicherstelle $B7 enthält die Länge des Dateinamens, hier erst einmal 0.

```
JSR   GETCHRERR
CMP   #'"'
BNE   LSERROR
```

Überprüft, ob Anführungsstriche eingegeben wurden. Falls nicht, springt unser Programm in die Routine »LSERROR« und bricht ab.

```
LS1   JSR   GETCHRERR
      STA   ($BB),Y
      INY
      INC   *$B7
      CMP   #'"'
      BNE   LS1
```

In diesem Programmteil wird der Filename eingelesen und in die Adresse gespeichert, die in $BB/$BC enthalten ist ($0201). Gleichzeitig wird $B7 als Zähler für die Namenlänge so lange erhöht, bis das zweite Anführungszeichen auftaucht. Damit ist der Filename gespeichert, beginnend bei $0201.

```
DEC   *$B7
LDA   IO.NR
STA   *$BA
LDA   *COMMAND
CMP   #'S'
BEQ   SAVE
```

Da die Namenlänge um eins zu groß geraten ist (das letzte Zeichen war ein »"«), muß sie dekrementiert werden. Die gewählte oder voreingestellte I/O-Nummer (Device-Nummer) soll in $BA gespeichert werden, damit die Betriebssystemroutine nachher das richtige Gerät anspricht. Zum Abschluß überprüft der Compare-Befehl, ob das Kommando »S« gewählt ist, um dann dorthin zu verzweigen.

```
LOAD   JSR   GETRET
       BEQ   LOAD1
       LDX   #$C3
       JSR   GETADR
       LDA   #$00
       STA   *$B9
```

Wir sind nun an der Stelle des Befehls angelangt, an der sich herausstellen muß, ob das Programm an seine Originaladresse (absolut) oder an eine andere Adresse geladen werden soll. Die Unterroutine »GETRET« prüft, ob unmittelbar nach dem Namen ein »RETURN« folgt und führt eine Verzweigung nach »LOAD1« durch, falls dieses eintritt. Ansonsten holen wir uns die Adresse und laden das vorgesehene Programm dorthin, nachdem in Speicherstelle *$B9 eine Null gespeichert ist, da ein absolutes Laden nicht erfolgt. Die Routine »GETADR« ist so aufgebaut, daß sie die nächsten 2 Bytes an die mit dem X-Register gewählte Stelle in der Zeropage ablegt. Dann führen wir ebenfalls »LOAD1« durch.

```
LOAD1   LDA   #$00
        JMP   ($0330)
SAVE    LDX   #$C1
        JSR   GETADR
        LDX   #$AE
        JSR   GETADR
        JMP   ($0332)
```

In »LOAD1« erfolgt der indirekte Sprung über $0330 in die LOAD-Routine des Betriebssystems.

Die SAVE-Routine erfragt vorher noch die fehlenden Adressen (Anfangs- und Endadresse des Programmes, das gespeichert werden soll), speichert sie nach $C1/C2 und $AE/AF und springt dann in die SAVE-Routine. Noch ein Wort zu den angesprochenen Betriebssystem-Routinen: Mittlerweile gibt es für den C 64 mindestens drei verschiedene Versionen des Betriebssystems von Commodore. Es sind zwar meist nur kleine Änderungen, aber die können fatale Folgen haben, wenn sich die Einsprungadressen ändern. Deshalb gibt es einen besonderen Bereich, das KERNAL, der einen Sprungverteiler für die wichtigsten Routinen enthält. Dieser wird grundsätzlich nie geändert. Beziehen Sie deshalb Ihre Einsprungadressen immer auf die KERNAL-Routinen, um sicher zu sein, daß Ihr Programm auch noch mit der zwölften Version des Betriebssystems läuft. Die KERNAL-Einsprünge stehen ganz hinten ab $FF81 im Speicher.

Als zweites ein Vergleich, der in Maschinenprogrammen häufig und in allen Variationen auftaucht: Es handelt sich dabei um den Vergleich zweier Adressen. Nun sind Adressen leider 16-Bit-Werte, unser Prozessor aber kann nur 8 Bit auf einmal verarbeiten. Gehen wir einmal von folgenden Bedingungen aus: Ein Programm soll von $4000 bis $4020 gelistet werden. Die Zeiger für das Ende befinden sich in Speicherstelle ENDLO (Lowbyte) und ENDHI (Highbyte). »PCL« (Programm-Counter-Low) und »PCH« (Programm-Counter-High) geben den augenblicklichen Stand des Programmes an. Dann erfolgt die Abfrage auf erreichtes Ende mit dieser Befehlsfolge:

```
VERGL   LDA   PCL
        CMP   ENDLO
        LDA   PCH
        SBC   ENDHI
        BCS   FERTIG
WEITER  JMP   AUSGABE
FERTIG  JMP   ENDE
```

Solange PCL und PCH kleiner sind als die Endwerte geht das Programm »WEITER«.

Sobald aber PCL und PCH die Werte von ENDLO und ENDHI erreicht haben, wird das Carry-Flag gesetzt und die Abfage mit BCS FERTIG würde das Auflisten anhalten. Daß es bei der Anwendung einige Probleme geben kann, sieht man daran, daß die Ausgabe auch schon unterbrochen wird, wenn gerade erst das Programmende erreicht ist. (Der letzte Befehl könnte »unter den Tisch fallen«.) Aber kein Problem ohne Problemlösung — und natürlich ohne weitere Probleme, die Sie aber mit ein bißchen Nachdenken sicher selbst lösen können.

Hinweise zum Abtippen

Tippen Sie das Ladeprogramm sorgfältig ab, speichern Sie es (!) und starten Sie mit RUN. Sollte es sich wider Erwarten auf Anhieb mit READY melden, haben Sie das Schlimmste geschafft. Ansonsten beseitigen Sie nun alle Fehler bis es zum READY durchläuft. Jetzt laden Sie das alte Ladeprogramm aus der letzten Ausgabe und starten es. Nach dem READY starten Sie SMON mit SYS 49152. Als erstes probieren Sie nun den Befehl »S«, um SMON selbst abzuspeichern, diesmal nicht mehr als Basic-Lader, sondern als Maschinenprogramm.

S "SMON $C000" C000 CAB7

SMON belegt jetzt 11 Blöcke auf der Diskette. Ab jetzt können Sie SMON direkt mit »LOAD "SMON $C000",8,1« laden und mit SYS 49152 starten.

Noch zwei Hinweise in eigener Sache: Einige wenige (!) Leser haben uns darauf aufmerksam gemacht, daß die Druckerausgabe auf bestimmten (exotischen) Druckern bisweilen kleinere Unzulänglichkeiten aufweist. Kurz und schlecht, uns ist in der letzten Folge ein Programmierfehler unterlaufen: Beim Disassemblieren verschwindet die letzte Zeile vor dem Strich (----) im Drucker und ward nicht mehr gesehen. So etwas passiert, wenn man kurz vor Redaktionsschluß noch auf die Schnelle kleine »Verbesserungen« vornimmt.

Für die Korrektur ist folgendes notwendig: Listen Sie mit »M C56C C57B« zwei Zeilen, gehen mit dem Cursor in die betreffenden Zeilen und geben folgende Änderung ein:

:C56C	**09**	C9	**30**	F0	**05**	C9	21	**DO**
:C574	**11**	**EA**	**20**	**94**	**C4**	20	51	C3

Nur die fetten Werte müssen geändert werden, alle anderen können Sie stehen lassen. Denken Sie bitte bei jeder Änderung daran, daß Sie die Zeile nur mit Drücken der RETURN-Taste an den Computer übergeben. Zur Probe können Sie ja noch einmal listen...

Wir haben nach dem letzten Artikel eine Menge Anrufe erhalten, von Lesern, die größtenteils Schwierigkeiten beim Eintippen der DATAs beziehungsweise bei der Fehlersuche hatten. Deswegen hier Hinweise zu den häufigsten Problemen:

1. Wenn nach Beendigung der Tipparbeit nach dem RUN eine Fehlermeldung ».... ERROR in 40« (oder 70) erfolgt, dann ist sicherlich nicht die Zeile 40 oder 70 daran schuld, sondern Sie haben aller Wahrscheinlichkeit nach einen Wert (ein »Datum«) falsch eingetippt. Der Computer bringt eine Fehlermeldung, wenn er beim POKE-Befehl auf eine Kommazahl trifft oder einen anderen nicht POKEbaren Wert. Dafür gibt es — neben schlichten Tippfehlern — mehrere Möglichkeiten: Es kann ein Komma fehlen oder durch einen Punkt ersetzt worden sein. Gerade dies ist nämlich auf dem Bildschirm sehr schlecht zu erkennen.

2. Überprüfen Sie nach dem Programmabbruch anhand des Direktbefehls »PRINT I« in welchem Block (+1) der Fehler steckt. Also bei der Antwort »1« steckt der Fehler in Block 2.

3. Der Direktbefehl »PRINT A« zeigt Ihnen den Wert, der den Fehler verursacht hat.

Versuchen Sie es erst einmal mit dieser kleinen Hilfe. Übrigens ist unser Listing mit 99prozentiger Wahrscheinlichkeit fehlerfrei, von uns und der Redaktion mehrfach durchprobiert. Das Dreckfuhlerteufelchen hat kaum eine Chance, da das Listing direkt von der Diskette auf den Drucker läuft.

Ich hoffe, daß Sie bis jetzt nicht in Ihren Bemühungen nachgelassen haben, möglichst häufig die verschiedensten Befehle zu probieren. Sie wissen doch: Nur die Übung macht den Meister — und das gilt speziell für die Maschinensprache. In der nächsten Ausgabe bekommen Sie dann die letzten Raffinessen des SMON, der dann komplett ist.

(Norfried Mann/gk)



```
10 rem ***************************
20 rem *                         *
30 rem *     smon                *
40 rem * von n.mann & d.weineck *
50 rem *      fleetrade 40       *
60 rem *      2800 bremen        *
70 rem *    tel. 0421 / 493090   *
80 rem *         0421 / 231401   *
90 rem *                         *
100 rem ***************************
110 fori=0to2:reada:pr(i)=a:next
120 sa=51261:i=0
130 pa=sa+256*i:ch=0
140 forj=0to255:reada:pokepa+j,a:ch=ch+a
:next
150 remifch<>pr(i)then200
160 i=i+1:ifi<2then130
170 pa=pa+256:ch=0
180 forj=0to121:reada:pokepa+j,a:ch=ch+a
:next
190 ifch=pr(i)thenend
200 print"fehler in block"i+1:end
210 rem
220 rem *** blockpruefsummen ***
230 rem
240 data34652,35523,16258
250 rem
260 rem *** block 1 ***
270 rem
280 data32,141,194,141,175,2,96,32,141,1
94,141,176,2,96,76,209,194,160,2,132
290 data188,136,132,185,132,187,136,132,
183,32,202,194,201,34,208,234,32,202
300 data194,145,187,200,230,183,201,34,2
08,244,198,183,173,176,2,133,186,165
310 data172,201,83,240,19,32,194,194,240
,9,162,195,32,128,194,169,0,133,185
320 data169,0,108,48,3,162,193,32,128,19
4,162,174,32,128,194,108,50,3,32,126
330 data194,32,202,194,73,2,74,74,8,32,1
28,194,32,81,195,40,176,12,165,253
340 data101,251,170,165,254,101,252,56,1
76,9,165,251,229,253,170,165,252,229
350 data254,168,138,132,252,133,251,132,
98,133,99,8,169,0,133,211,32,117,198
360 data165,252,208,15,32,73,195,165,251
,32,42,195,165,251,32,208,195,240,3
370 data32,35,195,32,76,195,162,144,165,
1,141,177,2,169,55,133,1,40,32,73,188
380 data32,221,189,174,177,2,134,1,76,86
,195,32,141,194,170,164,211,177,209
390 data73,32,240,163,138,168,32,154,194
,56,176,169,32,184,194,160,8,72,32
400 data202,194,201,49,104,42,136,208,24
5,240,235,32,184,194,162,0,138,134
410 data251,133,252,168,32,207,255,201
420 rem
430 rem *** block 2 ***
440 rem
450 data58,176,132,233,47,176,4,56,76,19
6,200,133,253,6,251,38,252,165,252
460 data133,254,165,251,10,38,254,10,38,
254,24,101,251,8,24,101,253,170,165
470 data254,101,252,40,105,0,76,52,201,3
2,122,194,169,55,133,1,162,4,189,135
480 data192,149,170,202,16,248,32,81,195
,166,170,165,171,32,205,189,230,170
490 data208,2,230,171,169,68,32,210,255,
169,193,32,210,255,160,0,177,251,132
500 data98,133,99,32,209,189,32,99,196,1
62,3,176,10,169,44,166,211,224,73,144
510 data227,162,9,134,198,189,125,192,15
7,118,2,202,208,247,76,110,195,32,122
520 data194,32,141,194,162,0,129,251,72,
32,99,196,104,144,247,96,32,90,194
530 data165,166,208,2,198,167,198,166,32
,48,202,134,181,160,2,144,4,162,2,160
540 data0,24,165,166,101,174,133,170,165
,167,101,175,133,171,161,164,129,168
550 data65,168,5,181,133,181,165,164,197
,166,165,165,229,167,176,29,24,181
560 data164,121,107,192,149,164,181,165,
121,108,192,149,165,138,24,105,4,170
570 data201,7,144,232,233,8,170,176,207,
165,181,240,15,76,209,194,56,162,254
580 data181,170,245,166,149,176,232,208,
247,96
590 rem
600 rem *** block 3 ***
610 rem
620 data32,98,202,76,214,201,76,98,202,1
97,167,208,2,228,166,176,19,197,165
630 data208,2,228,164,144,11,133,180,138
,24,101,174,170,165,180,101,175,96
640 data32,90,194,32,122,194,32,48,202,3
2,203,196,200,169,16,36,171,240,38
650 data166,251,165,252,32,70,202,134,17
0,177,251,133,181,32,74,197,160,1,32
660 data70,202,202,138,24,229,170,145,25
1,69,181,16,25,32,81,195,32,35,195
670 data36,171,16,15,177,251,170,200,177
,251,32,70,202,145,251,138,136,145
680 data251,32,106,198,32,102,196,144,18
1,96

ready.
```

Listing »SMON Teil 2« (Basic-Lader)

Mitmachen, aber wie?

Wie schicke ich meine Programme ein?

Die 64'er-Redaktion freut sich über jeden Beitrag unserer Leser. Die Erfahrung hat aber gezeigt, daß viele Einsender nicht genau wissen, in welcher Form sie ihre Manuskripte einsenden sollen. Die unten aufgeführten Punkte stellen keine »Richtlinien« dar. Dennoch sollte sich jeder, der ein Programm oder einen Artikel einsenden will, an ein gewisses Schema halten. Dies erleichtert zum einen die Arbeit der Redaktion, zum anderen kommt es auch Ihnen selbst zugute, da wir vollständige Listings oder Artikel schneller veröffentlichen können. Folgende Kriterien sind also generell zu beachten.

1. Auf der ersten Seite des Anschreibens sollten der Name, die vollständige Anschrift mit Telefonnummer sowie das Einsendedatum stehen.

2. In der »Betreffzeile« tragen Sie die genaue Spezifikation des verwendeten Computers und falls erforderlich, die Basic-, ROM- oder DOS-Versionen sowie die Speicherkonfigurationen ein. Der Titel des Artikels sollte ebenfalls daraus ersichtlich sein (auch für eventuelle Nachträge).

3. Im darauffolgenden Text können Sie Wesentliches zu Ihrer Person, zur Entstehungsgeschichte des Programms/Artikels, der Absicht, der Vorteile gegenüber anderen Programmen oder Methoden, der Eigenschaften und so weiter erläutern.

4. Auf der nächsten Seite beginnt die eigentliche Programmbeschreibung. Diese sollte nach Möglichkeit mit der Schreibmaschine geschrieben werden oder als Computerausdruck vorliegen. Den Text bitte mit mindestens eineinhalb oder doppeltem Zeilenabstand verfassen. Am linken und rechten Rand mindestens drei Zentimeter Freiraum für Korrekturen und Bemerkungen lassen.

5. Diese und alle nachfolgenden Seiten sollten durchnumeriert sein und in der Kopfzeile jeweils den Titel des Programms und den Namen des Autors enthalten.

6. Der Überschrift des Artikels schließen sich zwei oder drei einleitende Sätze an, welche die wesentlichen Punkte des Textes zusammenfassen.

Der Text selbst sollte in etwa folgenden Aufbau aufweisen:

— Angaben auf welchem Computer das Programm lauffähig ist sowie welche Erweiterungen und Peripherie notwendig sind

— ausführliche Beschreibung der Programmfunktion (mit Verweisen auf Ein-/Ausgabebeispielen wie Grafiken, Bildschirmfotografien, Hardcopys oder Diagrammen)

— detaillierte Programmbeschreibung (mit Verweisen auf Programmablaufplan, Variablendefinition, Startadressen der einzelnen Unterprogramme, Beschreibung wichtiger Programmzeilen etc.)

— eventuelle Umsetzung auf andere Basic-Dialekte oder Computer

7. Die genauen Lade- und Abspeicherschritte des Programms und der im Programm vorkommenden Routinen sollten dokumentiert sein.

8. Listings aus reprotechnischen Gründen nur als Original (keine Kopien) auf weißem, unliniertem Papier mit neuwertigem Farbband gedruckt einsenden. In den Listings dürfen grundsätzlich keine handschriftlichen Eintragungen stehen.

9. In den Kopfzeilen des Programms bitte den Titel desselben, die Computerkonfiguration, den eigenen Namen und die Adresse mit Telefonnummer eintragen (es soll vorkommen, daß sich Listings und Manuskripte verselbständigen, und mit beiden allein läßt sich wenig anfangen).

REM-Zeilen im Programm dienen der Übersichtlichkeit und sollten, falls nicht speicherkritische Aspekte dagegensprechen, immer zur Strukturierung eingesetzt werden (siehe u. a. »Sauberes Programmieren«).

10. Um das Eintippen für andere zu erleichtern, sollten Sie CHR$(X)-Werte und TAB(X) oder SPC(X) anstatt Cursor-Manipulationen für die Ausgabeformatierung verwenden. So ist die Befehlssequenz FOR I=1 TO 6:PRINT:NEXT zur Erzeugung von sechs Carriage Returns leichter einzutippen und auf andere Basic-Computer wesentlich einfacher zu übertragen. Und ist es nicht auch übersichtlicher statt einem Dutzend Cursor-Rechts-Symbolen einfach SPC(12) zu benutzen? Überprüfen Sie Ihr Programm einmal hinsichtlich dieser »Kleinigkeiten«.

11. Da wir (in Ihrem eigenen Interesse) nur getestete Programme veröffentlichen wollen, legen Sie bitte unbedingt eine Diskette oder Kassette, auf der das betreffende Programm mit mindestens einer Sicherheitskopie abgespeichert ist, bei. Auf der Diskette/Kassette und deren Umhüllung unbedingt den Namen mit vollständiger Adresse und Computerbezeichnung vermerken.

12. Wollen Sie mehrere Programme/Artikel gleichzeitig einsenden, so trennen Sie die Programme/Artikel nach dem oben aufgezeigten Schema. Die Einsendung mehrerer Disketten/Kassetten ist hingegen nicht notwendig.

13. Artikel können beliebig lang sein — von einzeiligen Routinen bis zu Serien über mehrere Ausgaben. Ein durchschnittlicher Artikel hat rund vier bis acht Schreibmaschinenseiten.

14. Hardcopys, Flußdiagramme, Zeichnungen und Bildschirmfotos dienen der Anschaulichkeit. Sie sollten nach Möglichkeit nicht fehlen. Zu jedem der vorgenannten »Zugaben« gehört aber eine Bildunterschrift und ein Verweis im Text.

15. Programme/Artikel die unserem Verlag zur Veröffentlichung angeboten werden, sollten aus urheberrechtlichen Gründen nicht gleichzeitig einem anderem Verlag vorliegen.

16. Das 64'er Magazin zahlt für Listings eine Pauschale zwischen 100 und 300 Mark. Für reine Artikel beträgt das Honorar zwischen 0,80 und 1,00 Mark pro Druckzeile. Für Disketten/Kassetten werden 30 Mark extra berechnet.

17. Sollten sich nach Erhalt eines positiven Anwortschreibens noch irgendwelche Änderungen oder Verbesserungen des Programms ergeben haben, teilen Sie uns das bitte umgehend mit. In diesem Falle benötigen wir ein vollständig neues Listing mit entsprechendem Datenträger.

(aa)

Bewegte Grafik und Text mischen

Wer an Sprites denkt, stellt sich meist kleine Männchen oder Raumschiffe vor. Aber auch beim Einsatz in Anwendungsprogrammen können Sprites zur grafischen Illustration sehr nützlich sein.

Als ich die Grafik entwarf, wollte ich ein Programm unter folgenden Gesichtspunkten schreiben: kein Spielkram und von Basic aus verwendbar. Damit diese Erweiterung auch mit anderen Basic-Erweiterungen ohne weiteres funktioniert, sollten die Befehle über »SYS« aufgerufen werden. Nun sollte es kein Allerweltsprogramm sein, sondern irgend etwas Neues. Da fielen mir die Sprites ein. Mit dem richtigen Programm könnten damit kleine aber deutliche Grafiken erstellt werden, zum Beispiel Niederschlags-Temperatur- oder Umsatztabellen. Als ideal fand ich vier Sprites hintereinander und das zweimal untereinander. Das gibt eine Auflösung von immerhin 95 x 42 = 4032 Punkten. Damit läßt sich schon einiges machen. Zur Handhabung werden vier Routinen benötigt. Eine zum Einschalten der Grafik, eine zum Setzen beziehungsweise Löschen der Punkte und eine zum schnellen Löschen der gesamten Grafik. Weiterhin noch eine, die die ganze Grafik auf eine Farbe setzt.

Das Einschalten und Positionieren der Grafik

Dazu sind vier Parameter nötig, zwei für die x,y-Koordinaten und zwei für die x,y-Ausdehnung, also das Spreizen der Sprites. Je nachdem, ob gespreizt oder nicht, werden die Koordinaten berechnet und in die jeweiligen Register abgelegt. Zum Schluß werden noch alle Sprites aktiviert. Der Einfachheit halber kann x nur Werte zwischen 0 und 255 haben. Bei x=255 und ungespreizt ragt die Grafik sowieso schon aus dem Bildschirm heraus. Auch sollte man mit dem y-Wert nicht zu hoch gehen, da sonst die Grafik möglicherweise oben in den Bildschirm hereinkommt. Diese Einschränkungen sind aber in den meisten Fällen unwesentlich.

Das Setzen beziehungsweise Löschen von einem Punkt

Dazu muß genau das Byte errechnet werden, in dem sich der Punkt befindet. Dafür braucht man nur die obere Spritereihe zu betrachten. Sie ist genau eine Speicherseite (256 Byte) groß. Sollte nun die y-Koordinate größer als 20 sein, so wird einfach eine Seite weitergeblättert, das heißt das Hi-Byte des Zeigers inkrementieren. Als erstes wird das Sprite errechnet. Es wird solange #24 (= eine Spritelänge) vom x-Wert abgezogen bis der x-Wert kleiner als 24 (24<x<=0) ist. So ist man praktisch im ersten Spriteblock gelandet. Nun wird vom x-Wert solange #8 abgezogen, bis der x-Wert kleiner als 8 (8<x>=0) ist. Jetzt weiß man, welches Bit gesetzt werden soll, und welches Byte in der Reihe gemeint ist (eins von dreien).

Nun der y-Wert. Ist er größer als 20, dann muß das Hi-Byte des Zeigers inkrementiert und vom y-Wert 21 abgezogen werden. Man will ja die Byteposition in einem Spriteblock errechnen. Ist der y-Wert jetzt noch größer als 0 so müssen noch einmal 3 * y dazugezählt werden, denn ein Sprite hat die Ausdehnung von 3 Byte. Damit hat man das Byte, das in einem Spriteblock verändert werden soll. Nun wird das Low-Byte des jeweiligen Sprites dazuaddiert und in der Zeropage für den Zeiger abgelegt. Nachdem das Bytemuster hergestellt ist, kann nun der Punkt gelöscht beziehungsweise gesetzt werden.

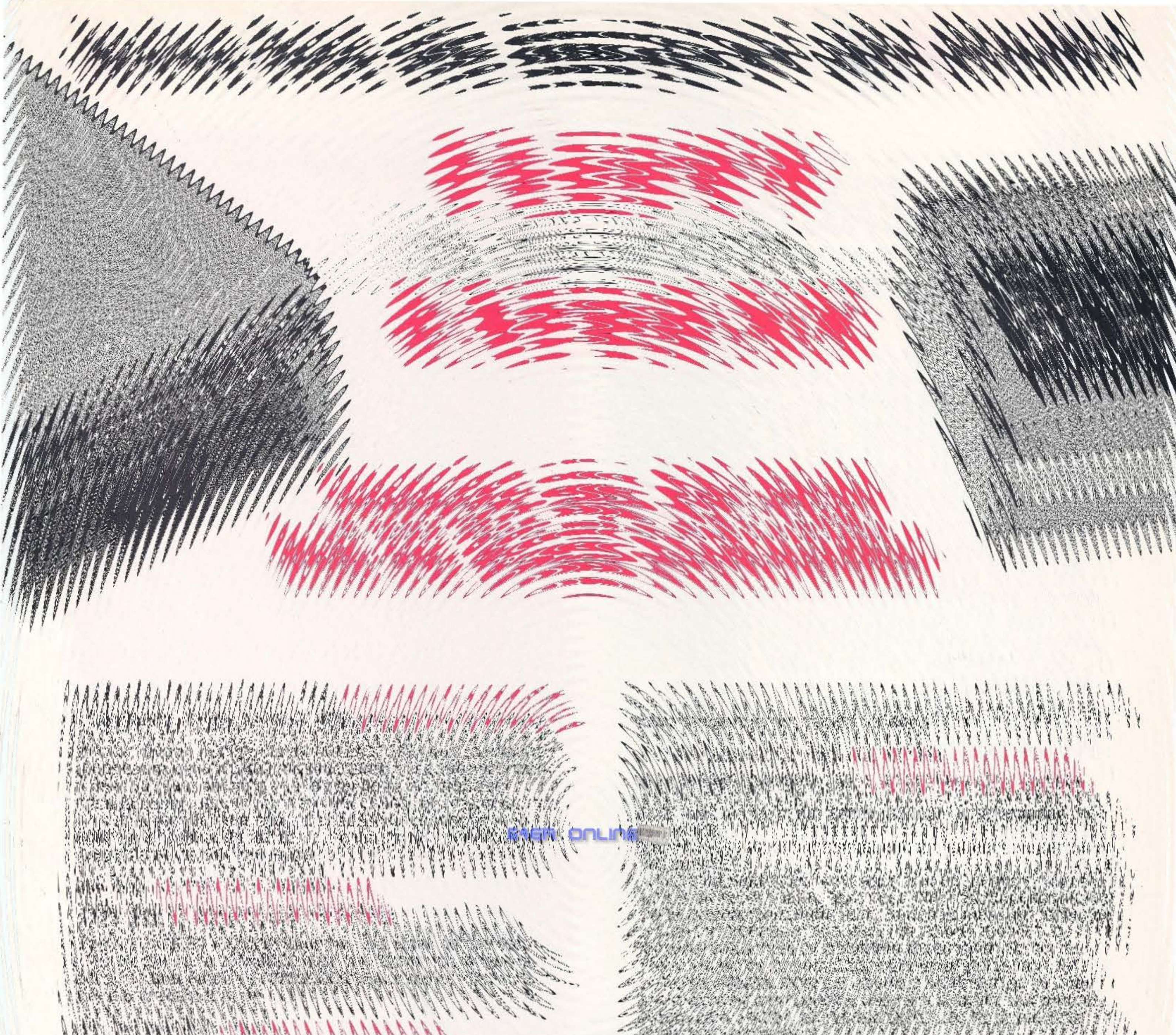
Das Löschen der Grafik und Setzen der Farbe versteht sich wohl von selbst. Das Ausschalten der Grafik geschieht durch POKE 53269,0.

Die Befehlsliste sieht dann so aus:

SYS PLOT , 0 , x-Wert , y-Wert	Punkt löschen
SYS PLOT , 1 , x-Wert , y-Wert	Punkt setzen
SYS SET , x-Pos. , y-Pos. , x-spreizen , y-spreizen	Grafik ein
SYS COLOR , Farbcode	Farbe setzen
SYS CLS	Grafik löschen

```
10 REM**********************************
20 REM*             S-GRAFIK           *
30 REM*       FREDERIC ESPITALIER      *
40 REM**********************************
50 FORA=49152TO49568:READX:POKEA,X
60 C=C+X:NEXTA
70 IF C<>52597THENPRINT"FEHLER IN DATAS"
:STOP
80 SYS49547
100 DATA32,253,174,32,158,183,224,96,48,
6,162,14,32,55,164,96,142,159,193,32
110 DATA253,174,32,158,183,224,42,16,237
,142,160,193,162,0,173,159,193,201
120 DATA24,48,6,56,233,24,232,208,246,14
1,159,193,169,8,133,188,189,155,193
130 DATA133,187,173,160,193,201,21,48,11
,230,188,173,160,193,56,233,21,141
140 DATA160,193,173,159,193,162,0,201,7,
48,8,240,6,56,233,8,232,208,244,41,7
150 DATA73,7,168,169,1,136,48,3,10,208,2
50,141,161,193,173,160,193,240,8,10
160 DATA24,109,160,193,141,160,193,138,2
4,109,160,193,24,101,187,133,187,96
170 DATA32,0,192,173,161,193,160,0,17,18
7,145,187,96,32,253,174,32,158,183
180 DATA224,2,48,3,76,10,192,224,1,240,2
26,32,0,192,173,161,193,73,255,160,0
190 DATA49,187,145,187,96,169,0,162,0,15
7,0,8,157,0,9,232,208,247,96,32,253
200 DATA174,32,158,183,138,162,8,157,38,
208,202,208,250,96,160,39,162,8,152
210 DATA157,247,7,136,202,208,248,32,253
,174,32,158,183,142,159,193,32,253
220 DATA174,32,158,183,142,160,193,32,25
3,174,32,158,183,224,2,48,3,76,10
230 DATA192,189,149,193,141,29,208,169,3
4,141,162,193,189,151,193,141,161
240 DATA193,162,0,142,16,208,173,159,193
,157,0,208,157,8,208,224,6,240,28,24
250 DATA109,161,193,205,159,193,176,11,7
2,173,16,208,13,162,193,141,16,208
260 DATA104,232,232,14,162,193,76,21,193
,32,253,174,32,158,183,224,2,48,3,76
270 DATA10,192,189,149,193,141,23,208,17
3,160,193,141,1,208,141,3,208,141,5
280 DATA208,141,7,208,24,125,153,193,141
,9,208,141,11,208,141,13,208,141,15
290 DATA208,169,255,141,21,208,96,169,0,
162,0,189,0,8,73,255,157,0,8,189,0,9
300 DATA73,255,157,0,9,232,208,237,96,16
9,10,133,44,141,130,2,76,148,227,0
310 DATA255,24,48,21,42,0,64,128,192,234
,0
READY.
```

Bild 1. Der Basic-Lader für die S-Grafik



SYS INV Grafik negativ
SYS INIT

Die Routine INIT bereitet das RAM auf S-Grafik vor. Sie schiebt den Speicher- und Programmstart auf $0A00, davor liegt jetzt der Speicher für die Sprites. Danach wird ein Kaltstart durchgeführt, das heißt der Basicspeicher gelöscht.

Technische Daten:
Sprites von $0800 bis $09FF
S-Grafik ab $C000 bis C1A0
Basic-RAM ab $0A00

Anwendung der S-Grafik:

Mit ein wenig Routine läßt sich mit der S-Grafik viel realisieren. Setzt man zum Beispiel die Farbe der oberen Spritereihe auf Rot und die der unteren Reihe auf Grün, so hat man einen roten und einen grünen Bereich. Nun kann man bei Säulengrafiken sofort erkennen, wo es kritisch wird (zum Beispiel Alkoholkonsum). Die Möglichkeiten sind fast unbegrenzt. Dazu kommt, daß man ja die Grafik überall hinschieben kann und auch noch spreizen kann. Sie kann im Textmodus beschriftet werden, im »Dunklen« aufbereitet werden, ruckzuck invertiert werden und so weiter.

Man lädt die S-Grafik mit LOAD"S-GRAFIK",8,1 (Bild 1). Nun kann S-Grafik ohne Bedenken eingesetzt werden.

Achtung! Wenn sich ein Basicprogramm im Speicher befindet, wird es durch Aufruf der Routine INIT gelöscht.

Der Grundgedanke zum Setzen eines Punktes.

Zum Beispiel (34/23)

A: 34 : 24 = 1 Rest 10

Der Punkt befindet sich in Sprite 1 und ist das 10 Bit

B: 10 : 8 = 1 Rest 2

Der Punkt befindet sich im 1. Byte, als 2. Bit

Y(=23) ist größer als 20, also wird Grundadresse 2 gewählt (Grundadresse 1 = $0800, Grundadresse 2 = $0900

C: 23-21 = 2

Das Byte befindet sich in der unteren Spritereihe in Reihe 2, das heißt 2*3 Byte müssen zur Adresse zugezählt werden.

Daraus ergibt sich:

64 Grundadresse von Sprite 0, Rechnung A
\+ 1 Byte aus Rechnung B
\+ 2*3 Byte aus Rechnung C
\+ $0900 Grundadresse der 2. Spritereihe
= Adresse des Bytes, in der sich der Punkt befindet.

Dazu ist noch zu bemerken, daß eine Spritereihe mit Sprite 0 beginnt. Wem das jetzt noch zu theoretisch ist, schaut sich einfach das Listing des Demo-Programms (Bild 2) an.

(F. Espitalier/rg)

```
1 poke53280,14:poke53281,6:print"▒"
10 plot=49299
11 cls=49331
12 farbe=49345
13 set=49361
14 inv =49523
20 sys cls
100 printchr$(14)"▒▒S-GRAFIK▒"
110 print"▒(C) by FCS 1984"
130 print"▒▒▒   Aufloesung    : 96*42 = 4
032 Punkte"
131 print"▒   Zoom            : je 2x nach
x und y"
132 print"▒   Position        : variabel"
133 print"▒   Farben          : variabel,ei
ne zur Zeit"
134 print"▒   Extras          : Grafik inve
rtieren"
136 forn=1to9999:geta$:ifa$=""thennext
140 sys set,100,100,1,1:print"▒":sys far
be,14
141 print"Ideal fuer Funktionsdarstellun
gen:"
150 forn=-20to20step1
160 y=0.1*n↑2:y=41-y
170 sys plot,1,n+45,y:next
171 forn=-20to20step1
172 y=0.1*n↑2:y=41-y
173 sys plot,1,n+48,y:next
174 forn=-20to20step1
175 y=0.1*n↑2:y=41-y
176 sys plot,1,n+51,y:next
177 forn=-20to20step1
178 y=0.1*n↑2:y=41-y
179 sys plot,1,n+54,y:next
190 sys inv:forn=1to9999:geta$:ifa$=""th
ennext
200 sys cls:sys farbe,13
201 print"▒Ideal fuer Mini-Grafiken"
210 print"▒▒Schnell erstellt, im Text Mo
dus"
211 print"▒Kann ueberall im Bildschirm s
tehen"
212 forx=0to95step8:ready:gosub 1000:nex
t
213 print"▒▒▒▒▒▒▒▒▒▒▒▒":forn=1to9999:get
a$:ifa$=""thennext
214 print"mal so,";
215 sys set,100,100,0,1
216 forn=1to9999:geta$:ifa$=""thennext
217 print"mal so";:sys set,100,100,1,0
218 forn=1to9999:geta$:ifa$=""thennext
219 print" oder so.":sys set,100,100,0,0
220 forn=1to9999:geta$:ifa$=""thennext
230 forn=0to255:sys set,n,100,0,0:forf=1
ton-200:nextf,n
240 sys set,100,100,1,1
250 forn=0to15:sys farbe,n:forf=0to400:n
extf,n
300 c=53248:print"▒"
310 sys cls
320 sys set,100,100,1,1
330 sys farbe,5:forn=43to46:pokec+n,2:ne
xt
340 poke53280,1:poke53281,1:print"▒"
350 forn=20to21:form=0to95:sys plot,1,m,
n:nextm,n
360 forx=0to4*13step13:read y:gosub500:n
ext
370 print"▒z.B. Gewinne Verluste bei ein
er Wahl"
380 print"▒▒▒▒▒▒▒▒Gewinne
     +        "
381 print"▒▒▒Verluste
     -        "
382 print"▒▒           A  B  C  D   E"
499 end
500 s=1:ss=(y*-1)+20:ifss<20thens=-1
501 for n=xtox+10:form=20tossteps:sys p
lot,1,n,m:nextm,n:return
1000 forn=xtox+6:form=41-yto41:sys plot,
1,n,m:nextm,n:return
2000 data2,10,8,17,22,41,33,29,18,1,3,7
3000 data +16,-9,+7,+12,-17

ready.
```

Bild 2. Ein Demo-Programm mit S-Grafik



Von allen Seiten betrachtet

Um dreidimensionale Körper von allen Seiten betrachten zu können, benötigen Sie den C 64, Simons Basic, einen Drucker und dieses Programm.

Das Programm soll einmal dreidimensionale Körper auf dem Bildschirm darstellen und zum anderen die hervorragende Grafikfähigkeit des C 64 demonstrieren.

Bei der Darstellungsart entschied ich mich für die normale Axonometrie. Sie ist einerseits leicht in eine für den C 64 verständliche Syntax zu packen und weist zum anderen einen räumlichen Effekt auf.

So entstand dann bald die erste Version von »Simons-Axo«. Allerdings war ich mit dieser Version noch nicht zufrieden. So tüftelte ich noch einige Routinen aus, die dem Programm erst den richtigen Schliff geben.

Nun fügten sich an die einfache Zeichenroutine noch weitere, die das Abspeichern der Körperdaten als sequentielle Datei auf Diskette, das Drehen des Körpers um die drei Koordinatenachsen (in beliebiger Variation), das Verschieben des Körpers, das Einzeichnen der Koordinatenachsen mit ihren Bezeichnungen und das Ausdrucken des Hires-Bildschirmes auf einen angeschlossenen Drucker ermöglichen.

Das Programm beginnt in der Zeile 10, wo der Bildschirm gelöscht und ebenso wie der Rahmen auf schwarz gesetzt wird. In den folgenden Zeilen gibt das Programm eine kurze Anleitung und wartet mit der Fortführung in Zeile 95, bis Sie eine beliebige Taste gedrückt haben.

In der Zeile 100 werden die Felder PT (Eckpunktkoordinaten, mit denen laufend gearbeitet wird) und PA (Eckpunktkoordinaten des Ausgangszustandes) mit 100,3 dimensioniert. Das bedeutet, daß Sie Körper mit bis zu 100 Eckpunkten eingeben können. Das Array ZP, ebenfalls mit 100 dimensioniert, gibt die Verbindungsvorschrift (Reihenfolge, in der die Punkte miteinander verbunden werden) an.

In Zeile 105—120 erfolgt eine Abfrage, ob die Körperdaten von der Diskette eingelesen werden sollen (Einleseroutine ab 1000) oder, ob Sie diese »von Hand« eingeben wollen. Sie können hier mit 'J' oder 'N' antworten, jede andere Antwort wird ignoriert.

Ab Zeile 125 steht die eigentliche Eingaberoutine, in der Sie mittels INPUT um die einzelnen Eckpunktkoordinaten gebeten werden. Diese werden den Feldern PT und PA zugeordnet. In dieser Eingabeschleife fungierten PX, PY und PZ als Zwischenvariablen und A als Zähler. Die Eingabe können Sie jeweils mit »Ende« abschließen. Hierbei empfiehlt es sich, den oben genannten Term einzufügen, da das Programm ja drei Variablen verlangt und sonst nur noch nachfragt, bis es alle drei hat. Mit »£« können Sie die vorige Eckpunkteingabe nocheinmal korrigieren (deswegen habe ich PX$ anstelle von PX verwendet).

Von 200 bis 270 wird die Verbindungsvorschrift eingelesen. Hier geben Sie zuerst den Startpunkt ein und dann jeweils einen weiteren Eckpunkt. Die Nummern dieser Eckpunkte werden nacheinander in ZP(B) abgelegt. In der Zeichenroutine werden die Eckpunkte dann in dieser Reihenfolge durch Linien verbunden, wodurch der Körper gezeichnet wird. Auch hier können Sie mit der »£«-Taste Korrekturen ausführen. In dieser Schleife dient B als Zähler und A$ als Zwischenspeicher. A1 enthält die Nummer des zuvor eingegebenen Eckpunktes (für die Korrektur notwendig).

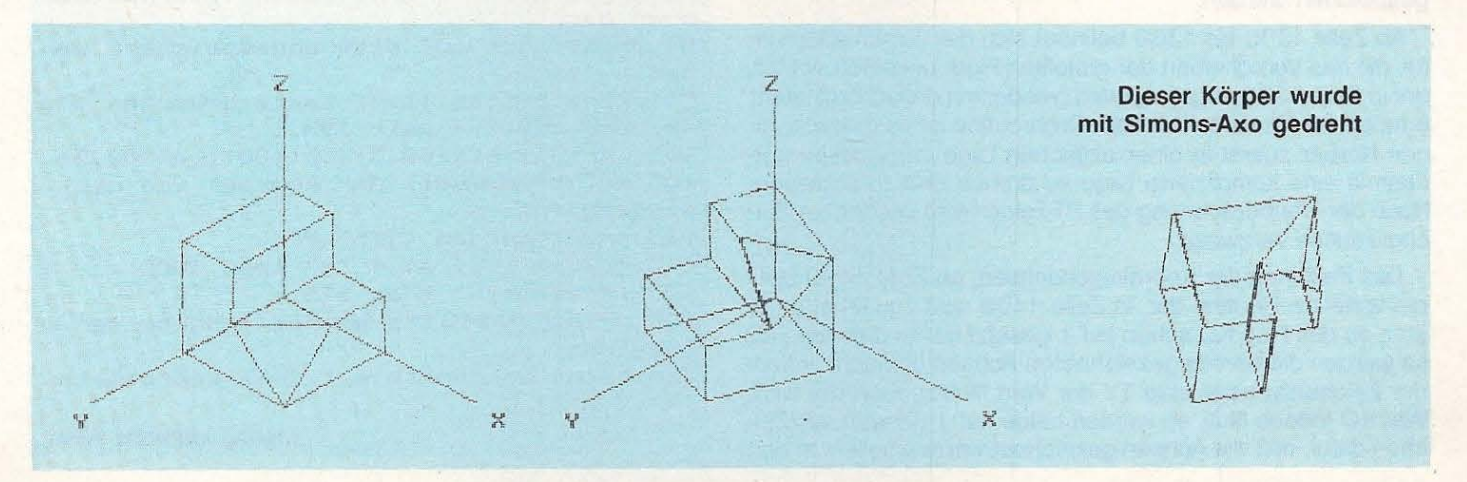
In 300 bis 360 können Sie das Koordinatenkreuz festlegen, indem Sie die Winkel zwischen der z- und y-Achse (AL) und z- und x-Achse (BT) eingeben. Das Programm ist für jeweils 120° voreingestellt, Sie können diese Vorschläge aber einfach überschreiben.

ACHTUNG: Sie wählen hier die Winkel zwischen den projezierten Koordinatenachsen. Die Winkel zwischen den realen Achsen im Raum sind natürlich immer je 90°!

Durch geeignete Wahl dieser Winkel können Sie den Sichtwinkel, unter dem Sie den dargestellten Körper betrachten, ändern. Sind beide Winkel 90°, so blicken Sie frontal von vorne auf den Körper, sind beide kleiner als 90°, so sehen Sie von unten her auf Körper und bei Winkeln über 90° von oben her.

Die nächsten Zeilen, von 400 bis 490 stellen die Zeichenroutine dar. Hier wird zunächst in Zeile 405 in den Hires-Modus umgeschaltet und die Zeichenfarbe grün bei schwarzem Hintergrund gewählt. In 410 bis 420 werden die Zeichenkoordinaten (zweidimensional) des Anfangspunktes, welcher durch ZP 0 gegeben ist, errechnet und in XA und YA abgelegt. In der nachfolgenden Schleife werden nach dem gleichen Schema die Endpunktkoordinaten für den LINE-Befehl des Simons Basic berechnet und die Linie gezeichnet. In Zeile 450 wird der Endpunkt nun zum Anfangspunkt für die nächste Linie definiert. Die Variable B dient hier als Zähler, während A als Index verwendet wird. Wenn alle Punkte miteinander verbunden sind, so wie es die Verbindungsvorschrift ZP(B) angibt, wird die Schleife beendet und das Programm prüft, ob das Flag für das Einzeichnen der Koordinatenachsen, KO auf 1 (zeichnen) gesetzt ist. Ist dies der Fall, so wird nach 1410 verzweigt.

In 480 und 485 wird in der linken oberen Ecke die Fertig-Meldung ausgegeben und in die Endlosschleife in 490 gesprungen, die nach Drücken der F1-Taste beziehungsweise RETURN verlassen wird.



Dieser Körper wurde mit Simons-Axo gedreht



Nun sind wir im Hauptmenü, welches uns zehn Möglichkeiten bietet. Die Abfrage erfogt in 585 mittels GET. In 590 steht die Sprungtabelle auf die einzelnen Routinen. Wenn Sie 0 eingegeben haben, wird das Programm nun in 595 abgebrochen. Mit GOTO XXXX oder CONT können Sie wieder einsteigen, ohne daß Sie die Daten verlieren.

In den Zeilen 600 bis 790 steht nun die Routine, die erlaubt, die erstellte Figur in beliebiger Variation um die Koordinatenachsen zu drehen. Doch zuerst gelangen Sie wieder in ein Menü, in dem Sie die momentane Drehung auswählen müssen. Der Drehwinkel ist in Grad einzugeben. Ist dieser positiv, wird die Figur gegen den Uhrzeigersinn um die gewählte Achse gedreht. Bei einem negativen Winkel erfolgt die Drehung im Uhrzeigersinn. Die Drehroutine ist in drei Abschitte unterteilt, je nach Drehachse, wobei alle drei mit den Polarkoordinaten arbeiten. Diese werden in den Zeilen 660 bis 670, 710 bis 720 sowie 760 und 766 berechnet. EO ist der Winkel und R der Radius. Zu EO wird nun einfach der Drehwinkel addiert und die neuen kartesischen Koordinaten berechnet. In diesen Schleifen dient X als Zähler, während EP die Anzahl der Eckpunkte angibt. In Zeile 790 wird wieder auf die Grafik zurückgeschaltet und in die Zeichenroutine nach 400 gesprungen. Nun kann man am Bildschirm wieder das Entstehen des Körpers in gedrehter Lage beobachten.

Die kurze Ausgangszustand-Routine in 800 bis 850 kopiert einfach die Anfangsdaten der Eckpunkte, die ja im PA-Feld festgehalten sind, in das Arbeitsfeld PT. Hier dienen X und Y wieder als Zähler. Danach wird die Zeichenroutine aufgerufen.

In den Zeilen 1000 bis 1097 steht die Einleseroutine, die die Körperdaten (Eckpunkte und Verbindungsvorschrift) von der Diskette, wo sie als sequentielle Datei stehen müssen, einliest und sie den entsprechenden Feldern zuordnet. Dem abgefragten Dateinamen, der auch den Joker enthalten kann, wird gleich »,S,R« angehängt und dann die Datei geöffnet, nachdem der Fehlerkanal geöffnet wurde (1010). Nun werden EP (Anzahl der Eckpunkte) und SP (letzte Nummer der Verbindungsvorschrift), die beiden wichtigen Steuervariablen, eingelesen. Danach werden zunächst die Felder PT und PA gefüllt und schließlich noch die Verbindungsvorschrift eingelesen und dem ZP-Feld zugeordnet. Zwischendurch wird in die Fehlerkanalroutine verzweigt, die sich in den Zeilen 1250 bis 1290 befindet und das Programm stoppt, wenn es zu einem Diskettenfehler kommt. Dann wird die komplette Fehlermeldung, die in den Variablen F1, F1$, F2 und F3 festgehalten ist, ausgegeben. Nun hat man die Möglichkeit (meistens, wenn man den Dateinamen falsch eingegeben hat) in die Zeile, in der der Fehler entstanden ist, zurückzukehren, indem man mit RETURN bestätigt, oder das Programm abzubrechen, indem man die SPACE-Taste betätigt.

Ab Zeile 1100 finden Sie die Schreibroutine. Diese funktioniert im wesentlichen genauso wie die Einleseroutine mit dem Unterschied, daß die momentanen Körperdaten auf Diskette gespeichert werden.

Ab Zeile 1300 bis 1360 befindet sich die Verschieberoutine, die das Verschieben der erstellten Figur um einen Vektor, der in Zeile 1320 abgefragt wird (wieder mit drei Koordinaten) ermöglicht. Mit dieser und der Drehroutine ist es möglich, einen Körper zuerst in einer einfachen Lage zu erstellen und dann in eine komplizierte Lage zu drehen und zu schieben. Nach der Neuberechnung des PT-Felder wird wieder zur Zeichenroutine verzweigt.

Das Zeichnen der Koordinatenachsen, ab Zeile 1400 stellt die vorletzte Routine dar. In Zeile 1405 wird zunächst getestet, ob das Flag KO schon auf 1 gesetzt ist. Ist dies der Fall, so werden die bereits gezeichneten Achsen gelöscht, indem der Zeichentypusvariable TY der Wert Null zugeordnet wird. War KO jedoch Null, so werden beide auf 1 gesetzt, als Zeichen dafür, daß die Achsen gezeichnet werden sollen. In den nächsten Zeilen werden den Variablen XS und YS die Endpunkte der Koordinatenachsen übergeben, während X1 und Y1 die Koordinaten zur Ausgabe der Bezeichnungen mit dem CHAR-Befehl darstellen. Es werden nun die Achsen nacheinander berechnet und dann gezeichnet und bezeichnet. Anschließend wird wieder auf den Grafikbildschirm umgeschaltet und nach 480 verzweigt, wo die Fertig-Meldung wieder ausgegeben wird.

In den Zeilen 1500 bis 1535 steht die letzte und gleichzeitig die einfachste Routine, das Ausgeben des Bildschirminhaltes auf einen angeschlossenen Drucker mit dem COPY-Befehl.

Vor der Hardcopy wird jedoch D$ mit »HARDCOPY« Meldung ausgegeben. Wer diese Meldung als störend empfindet, da sie auch auf dem Drucker ausgegeben wird, kann sie auch weglassen, indem er einfach die Zeilen 1515 und 1510 wegläßt. Ist die Hardcopy fertig, wird wieder nach 480 verzweigt und die Fertig-Meldung ausgegeben. (Peter Steger/rg)

```
1 rem ************************
2 rem * normale  axonometrie *
3 rem *                      *
4 rem *                      *
5 rem *     steger peter     *
6 rem *  bahnhofstrasse 20b  *
7 rem *  6632 ehrwald/triol  *
8 rem *  tel : a-05673/2656  *
9 rem ************************
10 : print"▒" : poke 53280,0 : poke 5328
1,0
20 printtab(10)"▒normale axonometrie :"
: print tab(10)"EEEEEEEEEEEEEEEEEEEEE▒"
25 print"▒▒▒▒▒dieses programm stellt  e
[illegible]laechig"
30 print"▒begrenzte koerper,nach eingabe
 bzw einle"
35 print"▒sen der eckpunkte und der verb
indungsvor"
40 print"▒schrift in normaler axonometri
e dar."
45 print"▒▒▒▒hierbei kann das achsenkreu
z beliebig"
50 print"▒gewaehlt werden."
55 print"allerdings  bleibt der ursprung
 in  der"
60 print"bildschirmmitte."
65 print"▒▒▒▒nach dem zeichnen der figur
 erscheint"
70 print"▒in der linken oberen ecke <fer
tig> und"
75 print"man kommt dann mit ▒<f1> oder <
return> ▒in"
76 print"▒das hauptmenue."
80 print"▒▒▒▒die erstellten  koerper koe
nnen dann"
85 print"auch auf disk abgespeichert wer
den."
90 print"▒▒"tab(11)"< taste druecken >";
95 poke 198,0 : wait 198,1
100 clr:dim pt(100,3),zp(100),pa(100,3)
105 print"▒▒▒▒soll der koerper von disk
eingelesen"
110 print"werden <j/n> ?"
115 get a$ : if a$ = "j" then 1000
120 if a$ <>"n" then 115
125 print"▒▒▒▒bitte um die eingabe der e
inzelnen"
126 print"eckpunkte mit drei koordinaten
 (x,y,z)"
```

Listing »Simons Axo«

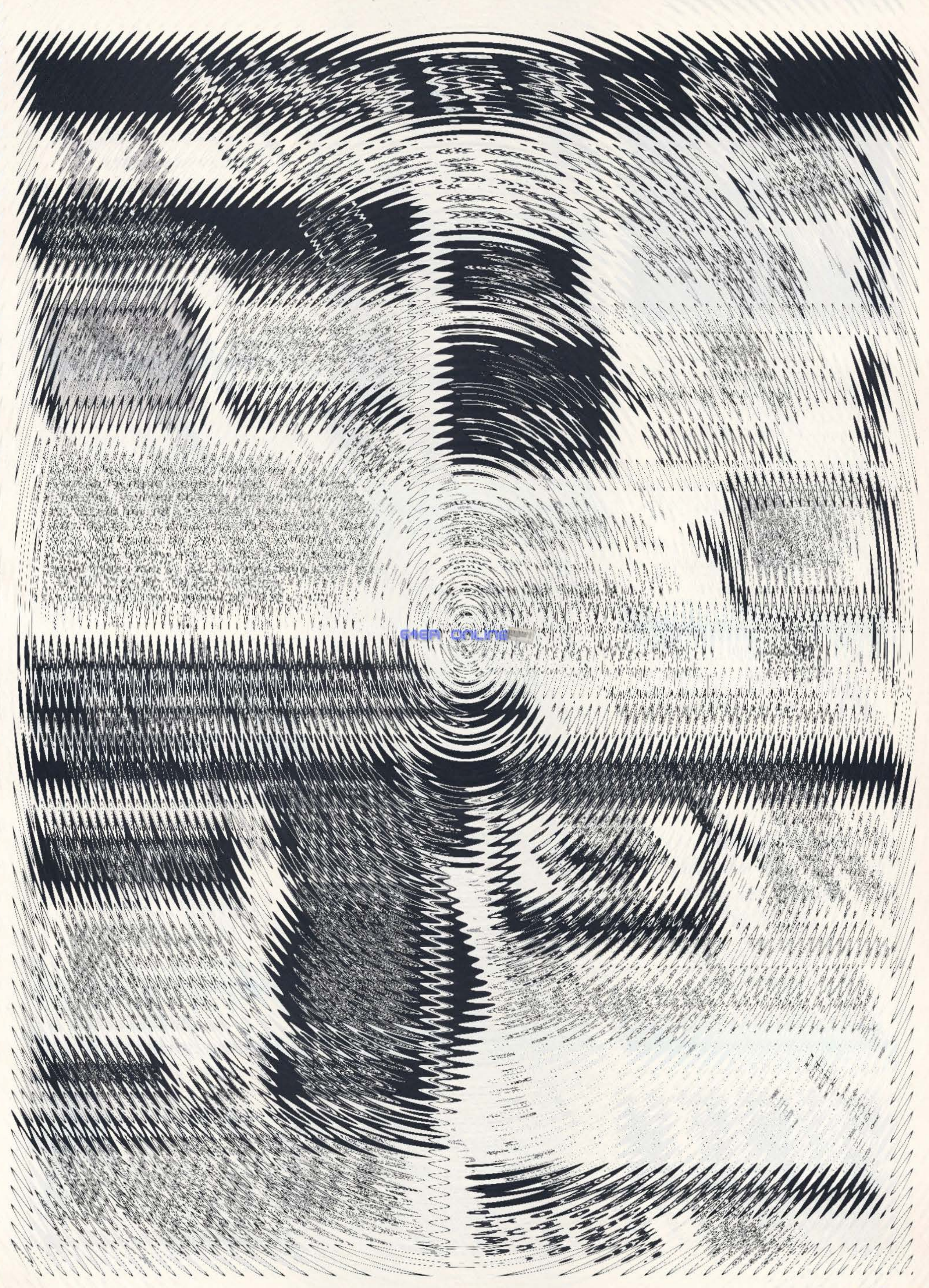
```
127 print"■bei x='ende' wird die eingabe
 beendet."
128 print"■bei x='£' kann der vorige pun
kt nochein"
129 print"mal eingegeben werden.■■"
130 print"("a")"; : input px$,py,pz : px
=val(px$) : if px$="ende" then 200
135 if px$="£" then a=a-1 : goto 130
140 pt(a,1) = px
145 pt(a,2) = py
150 pt(a,3) = pz
155 pa(a,1) = pt(a,1)
160 pa(a,2) = pt(a,2)
165 pa(a,3) = pt(a,3)
170 a=a+1 : goto 130
200 rem
201 rem verbindungsordnung
202 rem
205 ep = a-1 : print"■■■■in welcher reih
enfolge sollen welche"
210 print"punkte verbunden werden ?
215 print"■'ende' beendet die eingabe wi
eder."
220 print"■'£' laesst die korrektur der
vorigen "
225 print"eingabe zu."
230 input "■■ausgangspunkt";a
235 zp(0)=a
240 b=1 : print"■" : rem eingabeschleife
245 print"von ("a") nach : "; : input a$
 : if a$= "ende" then 270
250 if a$ = "£" then b=b-1 : a=a1 : goto
 245
255 a1=a : a = val(a$) :
260 zp(b) = a
265 b=b+1 : goto 245
270 sp=b-1
300 rem
301 rem koordinatensystem festlegen
302 rem
305 print"■■■■■■bestimmen sie nun das ac
hsenkreuz."
310 print"■■■■geben sie die winkel zwisc
hen :"
315 print"■■"
320 print tab(19)" |"
325 print tab(19)" |"
330 print " z- und y-achse ";tab(19)"NM"
;tab(24)"z- und x-achse "
335 print tab(18)"N  M"tab(24)"■■■■■ein.
"
345 print tab(17)"N    M"
350 rem
355 input "■ alpha =  120■■■■■";al:print
tab(25);:input"■  beta = 120■■■■■";bt
360 al=al*π/180 : bt=bt*π/180
400 rem
401 rem zeichenroutine
402 rem
405 hires 5,0
410 a=zp(0)
415 xa = 160 - sin(al)*pt(a,2) + sin(bt)
*pt(a,1)
420 ya = 100 - pt(a,3) - cos(al)*pt(a,2)
 - cos(bt)*pt(a,1)
425 for b = 1 to sp
430 a=zp(b)
435 xs=160-sin(al)*pt(a,2)+sin(bt)*pt(a,
1):if xs>319 then xs=319:ifxs<0thenxs=0
440 ys=100-pt(a,3)-cos(al)*pt(a,2)-cos(b
t)*pt(a,1):ifys>199 then ys=199
441 if ys<0 thenys=0
445 line xa,ya,xs,ys,1
450 xa=xs : ya=ys
455 next
460 if ko = 1 then 1410
480 d$="■■fertig>"
485 text 5,5,d$,1,0,7
490 get a$ : if a$ <> chr$(133) and a$ <
> chr$(13) then 490
500 rem
501 rem hauptmenue
502 rem
505 nrm : print"■"tab(10) "hauptmenue :"
510 print tab(10)"EEEEEEEEEEEE"
515 print"■■■■(1) = neues achsenkreuz
520 print"■■■■(2) = drehung der figur um
 oo
525 print"■■■■(3) = ausgangszustand hers
tellen"
530 print"■■■■(4) = verschieben der figu
r"
535 print"■■■■(5) = koordinatenachsen ei
nzeichnen"
545 print"■■■■(6) = abspeichern der koer
perdaten"
550 print"■■■■(7) = hardcopy auf drucker
555 print"■■■■(8) = neuer start"
560 print"■■■■(9) = zur grafik zurueck"
580 print tab(10) "■(0) = ende"
585 get a$ : if a$ <"0" or a$ >"9" or a$
="" then 585
590 on val(a$) goto 300,600,800,1300,140
0,1100,1500,100,1220
595 stop
600 rem
601 rem drehung der figur um oo
610 print"■■■■■■■drehung der figur um oo
 :":print"■■■■■EEEEEEEEEEEEEEEEEEEEEEEE"
615 print"■■■■■bitte waehlen sie aus :"
620 print"■■■■■(1) = drehung um die z-ac
hse.
625 print"■■■■(2) = drehung um die y-ach
se.
630 print"■■■■(3) = drehung um die x-ach
se."
635 get a :if a < 1 or a >3 then 635
636 z=5+2*a : inv z,3,29,1
640 print tab(7)"■■■bitte den drehwinkel
 :";:input de : de=de*π/180
645 on a goto 650,700,750
650 rem
651 rem um die z-achse
655 for x=0 to ep
660 eo=atn(pt(x,1)/(pt(x,2)-1e-32))-π*(p
t(x,2)<=0)-2*π*(pt(x,1)<0 andpt(x,2)>0)
665 eo = eo+de : if eo >= 2*π then eo =
eo-2*π
670 r=sqr(pt(x,1)↑2+pt(x,2)↑2)
675 pt(x,1)=sin(eo)*r
680 pt(x,2)=cos(eo)*r
685 next
690 goto 790 : rem return
700 rem
701 rem um die y-achse
705 for x=0 to ep
710 eo=atn(pt(x,3)/(pt(x,1)-1e-32))-π*(p
t(x,1)<=0)-2*π*(pt(x,3)<0andpt(x,1)>0)
715 eo=eo+de : if eo >= 2*π then eo=eo-2
*π
```

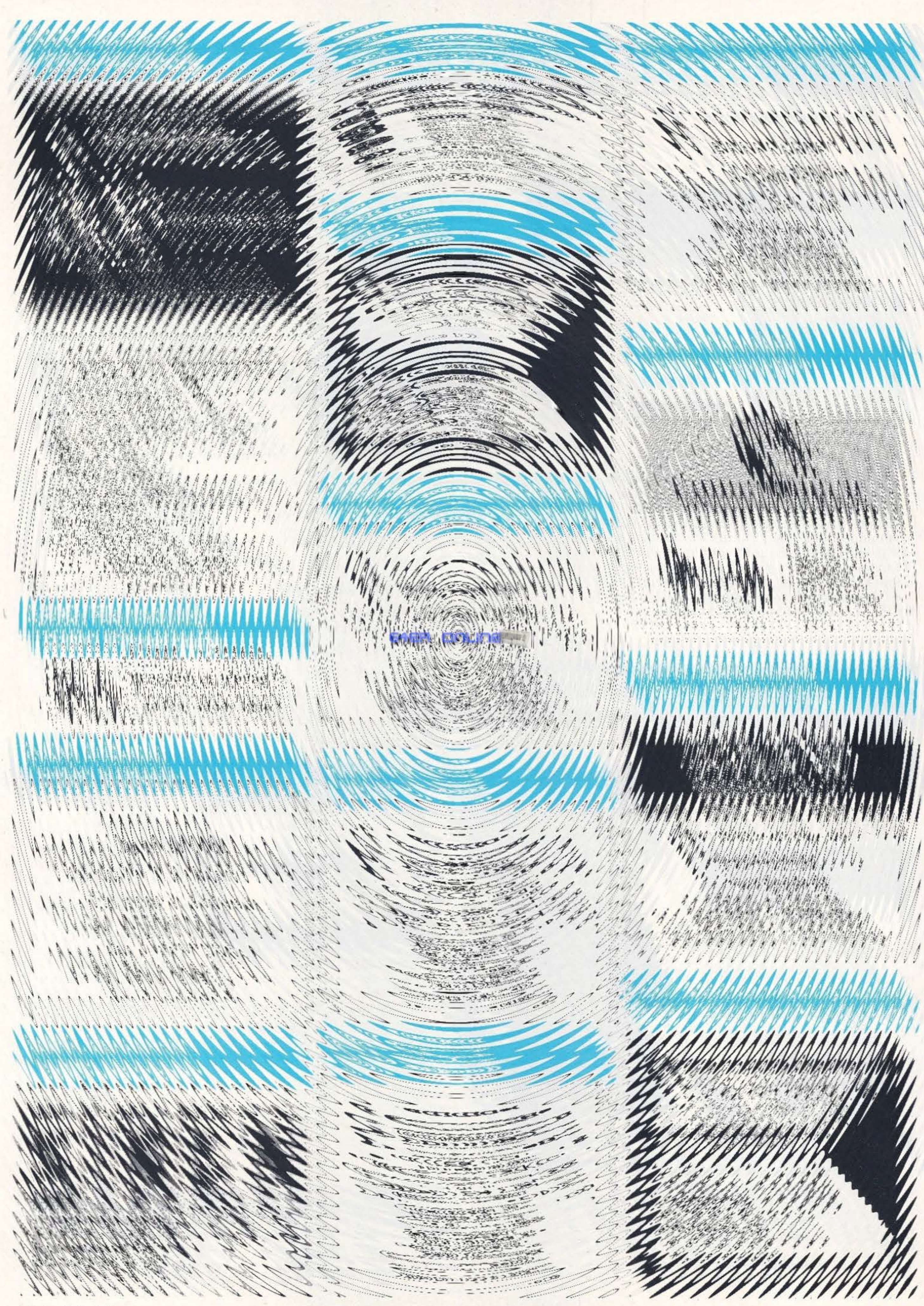
Listing »Simons Axo« (Fortsetzung)

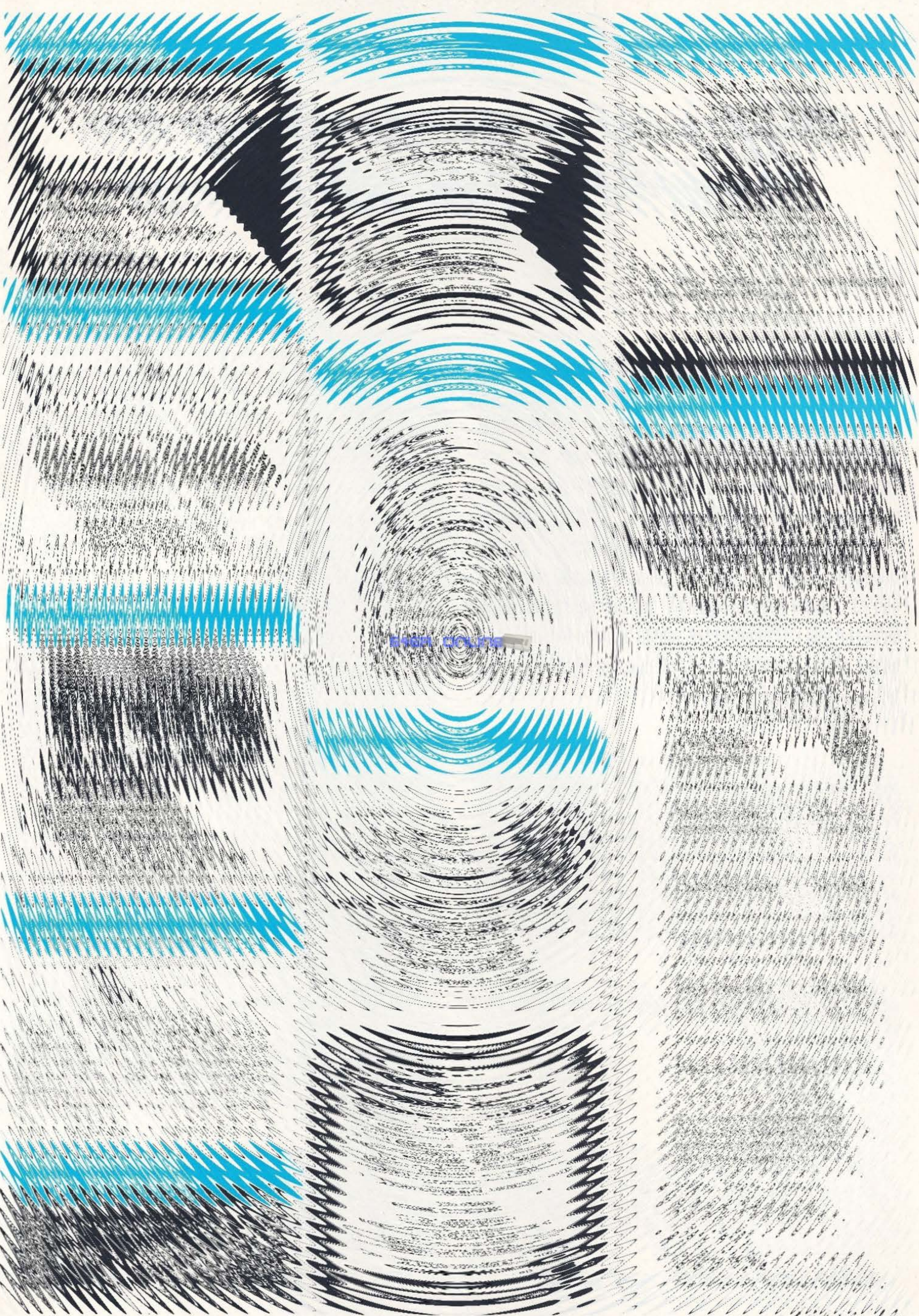
```
720 r = sqr(pt(x,1)↑2+pt(x,3)↑2)
725 pt(x,1)=cos(eo)*r
730 pt(x,3)=sin(eo)*r
735 next
740 goto 790 : rem return
750 rem
751 rem um die x-achse
755 for x=0 to ep
760 eo=atn(pt(x,2)/(pt(x,3)-1e-32))-π*(p
t(x,3)<=0)-2*π*(pt(x,2)<0andpt(x,3)>0)
765 eo=eo+de : if eo>= 2*π then eo = eo-
2*π
766 r = sqr (pt(x,2)↑2+pt(x,3)↑2)
770 pt(x,3)=cos(eo)*r
775 pt(x,2)=sin(eo)*r
780 next
790 cset 2 : goto 400 : rem return
800 rem ausganagszustand
801 rem
810 for x = 0 to ep
820 for y = 1 to 3
830 pt(x,y)=pa(x,y)
840 nexty,x
850 goto 400
999 rem
1000 rem daten vom der disk holen
1001 rem
1010 open 1,8,15 : rem fehlerkanal
1015 print"bitte dateinamen :"; : inpu
t ns$ : n$=ns$
1020 ns$=ns$+",s,r"
1025 open 2,8,2,ns$
1030 gosub 1250
1032 if f1 = 62 then close 2 : goto 1015
1035 print"einlesen der daten von :";n
$
1040 input#2,ep
1045 input#2,sp
1050 for x = 0 to ep
1055 for y = 1 to 3
1060 input#2,pt(x,y)   : pa(x,y)=pt(x,y)
1065 next y
1070 gosub 1250
1075 next x
1080 for x = 0 to sp
1085 input#2,zp(x)
1090 next x
1095 gosub 1250
1097 close 2 : close 1 : goto 300
1100 nrm : rem koerperdaten auf disk
1101 rem
1110 print "koerperdaten auf
 disk:"
1120 print"EEEEEEEEEEEEEEEEEEEEE
"
1130 print"bitte dateinamen :"; : inp
ut ns$
1140 open 1,8,15 : rem fehlerkanal
1150 ns$=ns$+",s,w"
1160 open 2,8,2,ns$
1162 print#2,ep : print#2,sp
1165 for x = 0 to ep : rem punkte
1170 for y = 1 to 3
1180 print#2,pt(x,y)
1185 next y
1190 gosub 1250
1195 next x
1200 for x = 0 to sp : rem verbindungsvo
rschrift
1205 print#2,zp(x)
1210 next
1215 gosub 1250
1220 close 2 : close 1 : cset 2 : goto 4
90
1250 rem
1251 rem fehlerkanal
1252 rem
1255 input#1,f1,f1$,f2,f3 : if f1 = 0 th
en return : rem kein fehler
1260 print "fehler auf der disk !"
1265 print""f1,f1$"  "f3"  "f4
1270 print"bitte mit <return> bestaeti
gen,oder mit"
1275 print"<space> das programm abbre
chen !"
1280 getan$ : if an$ = chr$(13) then ret
urn
1285 if an$ <> chr$(32) then 1280
1290 close 1 : stop
1300 rem
1301 rem verschieben der figur
1302 rem
1305 print"verschieben der figur
:":print"EEEEEEEEEEEEEEEEEEEEEEE"
1310 print"bitte um den verschiebe
vektor in der"
1315 print"form 'x,y,z' !"
1320 input"vektor :";px,py,pz
1322 print"bitte etwas geduld ."
1325 for a=0 to ep
1335 pt(a,1) = pt(a,1) + px
1340 pt(a,2) = pt(a,2) + py
1345 pt(a,3) = pt(a,3) + pz
1350 next a
1360 goto400
1400 rem
1401 rem koordinatenachsen
1402 rem
1405 if ko=1 then ty=0 : ko=0 : goto 141
5
1410 ko=1 : ty = 1
1415 xs=160-sin(al)*95 : x1=160-sin(al)*
105
1420 ys=100-cos(al)*95 : y1=100-cos(al)*
105
1425 char x1,y1,25,ty,1
1430 line 160,100,xs,ys,ty : rem y-achse
1435 xs=160+sin(bt)*95 : x1=160+sin(bt)*
102
1440 ys=100-cos(bt)*95 : y1=100-cos(bt)*
105
1445 line 160,100,xs,ys,ty : rem x-achse
1450 char x1,y1,24,ty,1
1455 line 160,100,160,10,ty : rem z=achs
e
1460 char 155,0,26,ty,1
1470 cset 2 : goto 480
1500 rem
1501 rem hardcopy
1502 rem
1505 text 5,5,d$,0,0,7
1510 d$="<hardcopy>"
1515 text 5,5,d$,1,0,8
1520 cset 2
1525 copy
1530 text 5,5,d$,0,0,8
1535 goto 480

ready.
```

Listing »Simons Axo« (Schluß)







Trace und Single Step für Maschinenprogramme

Maschinenprogramme stürzen bei Fehlern meist ohne Hinweis auf den Fehlerort ab. Mit Trace lassen sich Maschinenprogramme Befehl für Befehl abarbeiten. Es werden dabei die momentanen Registerwerte (Programmcounter, X-, Y-Register, Akku, Stackpointer und Flags) angezeigt. Diese Werte lassen sich ohne weiteres ändern. Als Dreingabe erscheinen auch noch der Zustand des User-Ports und des Datenrichtungsregisters B auf dem Bildschirm.

Das Programm läßt eine Ausführung von Maschinenprogrammen im Einzelschrittmodus zu. Dabei wird das Programm wirklich ausgeführt also nicht simuliert. Es bietet Einzelschritt, langsam und schnellen Trace. Alle Register werden angezeigt und können verändert werden. Zusätzlich wird der Befehl disassembliert. Laden und Abspeichern des Programms geschieht je nach Assemblerversion des Lesers. Das Assemblerprogramm erlaubt ein freies Verschieben von Trace durch Änderung in Zeile 100: *=$XXXX. Der Startwert für PC (Programmcounter) kann in Zeile 270 frei gewählt werden.

Programmfunktionen

Nach dem SYS-Befehl befindet man sich im Tracemodus. Es sind nur die Tasten F1 bis F7 und X für Exit aktiv. Folgende Register werden angezeigt:

1. PC = Programmcounter.
2. SP = Stackpointer.
3. YR = Y-Register.
4. XR = X-Register.
5. AC = Akkumulator.
6. Prozessorstatusflags.
7. User-Port mit Datenrichtungsregister. Entsprechende Eingangsleitungen sind revers dargestellt.
8. Es werden 1 bis 3 Hexbytes angezeigt, die verändert werden können, dann wird der Befehl disassembliert dargestellt.

Trace wartet jetzt auf einen Tastendruck. Der Programmcounter zeigt am Anfang auf Hex C000.

Folgende Tastenfunktionen stehen zur Verfügung:
Taste F7 = Einzelschritt. Pro Tastendruck wird ein Befehl des Testprogramms ausgeführt (genauer gesagt beim Loslassen von F1). Danach werden wieder die Register angezeigt und disassembliert.
Tast F5 = Slow Trace. Solange diese Taste gedrückt ist, wird das Testprogramm Befehl für Befehl abgearbeitet, die Register angezeigt und pro Befehl noch zusätzlich um zirka 65 ms verzögert.
Taste F3 = Tast Trace. Wie F5 jedoch ohne Verzögerung.
Taste F1 = Set Register, Editmodus. Durch Drücken von F1 kommen Sie in den Editmodus.

Beschreibung Editmodus

Die erste Ziffer des Programmcounters wird zur Kennzeichnung des Cursororts revers dargestellt. Sie können jetzt alle angezeigten Werte überschreiben. Der Stackpointer und die Userport-Anzeige können zwar überschrieben werden, dies hat aber keinen Einfluß auf die Werte. Eine Veränderung des SP würde meist zum Absturz des Systems führen, da die CPU nach dem Interrupt nicht mehr die korrekten Rücksprungadressen auf dem Stack vorfindet. Die Cursor-Right-Taste bewegt den Cursor nach rechts, die Cursor-Up-Taste nach links. Neben diesen Steuertasten sind nur die Tasten A bis F, 0 bis 9 für die Hexziffern, der »Pfeil nach oben«, um ein Flag zu setzen, und das »Minuszeichen«, um ein Flag zu löschen, sinnvoll. Die Spacetaste ist auch erlaubt.

Die DELETE-Taste funktioniert nicht. Entsprechen die Änderungen in der PC-Zeile Ihren Wünschen, dann drücken Sie ⟨RETURN⟩ und können die disassemblierte Zeile editieren. Hier sind je nach Befehlslänge 1 bis 3 Hexbytes am Anfang der Zeile veränderbar, das heißt Sie können die Operanden oder auch den Opcode selbst noch vor der Ausführung verändern, was zum Austesten von Programmen sehr praktisch ist. Verlassen können Sie die disassemblierte Zeile wieder mit ⟨RETURN⟩. Sie befinden sich jetzt wieder im Tracemodus, das heißt die Tasten F1, F3, F5, F7 und X sind wieder aktiv.

Den Tracemodus können Sie durch Drücken der X-Taste verlassen, der Computer ist jetzt im Basic-Editmodus. Trace kann mit SYS 49152 wieder gestartet werden, wobei hier der Disassembler immer aktiviert ist. Starten Sie mit SYS 49160, dann ist der Disassembler nur beim Editieren an, Trace ist dann etwas schneller. Setzen Sie Trace nicht auf sich selbst an.

Funktionsweise

Siehe hierzu auch Assemblersourcelisting. Zuerst schalte ich den normalen Tastaturinterrupt über Timer A aus und benütze den Timer B in CIA 1 für meine Interruptroutine. Timer A läuft zwar weiter, aber sendet keine IRQ mehr. Dann setze ich den IRQ-Vektor auf Trace. Trace initialisiert beim Start den Stack und beginnt dann mit der eigentlichen Einzelschrittroutine. Diese Routine dient als neue Interruptroutine, da der IRQ-

Vektor auf Trace geändert wurde. Der Interrupt wird aber nicht mehr alle $\frac{1}{60}$ Sekunde durch den Timer A ausgelöst, sondern durch Timer B und zwar alle 25 Mikrosekunden nach Start des Timers.

In dieser Zeit kann der Prozessor das Ende der normalen IRQ-Routine durchlaufen (Register vom Stack holen). Jetzt bleibt aber nur mehr Zeit, um einen Befehl des Testprogramms abzuarbeiten, da Timer B ja schon in genau einer Mikrosekunde erneut einen IRQ sendet. Der Prozessor legt nach diesem einen Befehl alle Register auf den Stack und verzweigt entsprechend dem IRQ-Vektor wieder auf Trace. Für genauere Information siehe Listing. Innerhalb der Traceroutine wird ein IRQ nicht akzeptiert, da sich sonst das Programm immer selbst unterbrechen würde. Trace läßt sich also nur durch einen NMI unterbrechen.

Beispiel:
SYS 49152 = Trace starten. Disassembler an.
Taste F1 drücken = Editmodus wählen.
PC auf $AF08 setzen. Diese Systemroutine gibt »Syntax Error« aus. Sie können zum Spaß auch die Register verändern, benützen Sie auch die Cursorsteuertasten.

RETURN drücken = Edit für disassemblierte Zeile. Hier können Sie die Hexbytes editieren. Eine Änderung hätte aber keinen Sinn, da wir ja eine ROM-Routine tracen. Drücken Sie nur ⟨RETURN⟩, und Sie gelangen wieder in den Tracemodus.
F7 drücken = Einen Befehl ab PC ausführen.
F5 drücken = Trace. Anzeige läuft. Flags werden in schneller Folge geändert. Die disassemblierte Zeile ändert sich sehr schnell. Halten Sie die Taste gedrückt, nach ein paar Sekunden sehen Sie schön langsam Buchstabe für Buchstabe »Syntax Error« erscheinen. Befindet sich der Cursor zufällig gerade am unteren Bildschirmrand und ist der Bildschirm vollgeschrieben, dann kann man schön verfolgen, wie die Scrollroutine arbeitet, der Bildschirminhalt wird Zeile für Zeile nach oben geschoben, um Platz für die Meldung zu machen.

(Jürgen Göbel/aa)

LABELTABELLE

BMINUS	C233	BYT	C1F6	L007	C2FB	L008	C311
CIA	DC00	CLE1	C27E	L009	C31A	LOOP	C1BF
COD1	C483	COD2	C484	LOOP1	C1D2	MREAD	C15F
COD3	C485	CODE2	C33D	OFFSET	C216	OLDX	C482
CODEL	C486	CODL3	C375	OLDY	C481	OPP	C325
COL	D800	CONT1	C079	PPOIN1	C2BE	PPOIN2	C2C4
CONT2	C090	CONT3	C0A4	PPOIN3	C2CA	PRBYTE	C10E
CONT4	C0E9	CONT6	C17D	PREAD1	C404	PREAD2	C422
CRB	DC0F	DDRB	DD03	PREAD3	C43D	PRIZEI	C12C
DISAS	C274	DISOFF	C05C	READ	C1A2	READPC	C3D5
DISON	C487	ENDAS	C3CE	READY	A473	RECHTS	C173
EREAD	C440	EXIT	C0F1	REGISTER	C063	REVERSE	C160
FAST	C0CF	FLSETZ	C081	SCREEN	0400	SET	C13E
GET	C138	GO	C00D	SLOW	C0E7	SPACE	C12A
HEX	C469	ICR	DC0D	STACK	0100	START	C143
IRQVEK	0314	LAENGE	C253	TADR	C800	TAST	EA87
LBIT	C209	LINKS	C169	TEXT	C441	TH	DC07
L001	C271	L0010	C34D	TITEL	C037	TL	DC06
L0011	C356	L0012	C365	TMP	C479	TMP1	C480
L0013	C381	L0014	C394	TRACE	C034	TSTART	C500
L0015	C3A5	L0016	C3BB	USETZ	C098	WAIT	C0AE
L002	C272	L003	C273	WAITL	C0C8	ZAHL	C213
L005	C2A4	L006	C2B8	ZEICHHOL	C182		

Listing von »Trace und Single Step«

```
2
100:   C000               *=    49152
                         ;
                         ;
                         ;TRACE / SINGLE STEP FUER C64        9.84
                         ;
                         ;   +DISASSEMBLER / EDITOR
                         ;
                         ;       JUERGEN GOEBEL
                         ;
                         ;       8 MUENCHEN 82
                         ;
                         ;       GROSCHENWEG 19
                         ;
                         ;
                         ;
                         ;      TEL. 089 / 432709
                         ;
                         ;
150:   C000                     .OPT OO,P1
160:   C000           IRQVEK    =    $314
160:   C000           STACK     =    $100
160:   C000           COL       =    55296    ;HINTERGRUNDFARBE
170:   C000           SCREEN    =    1024
170:   C000           READY     =    $A474-1
180:   C000           CIA       =    $DC00
180:   C000           TL        =    CIA+6    ;TIMERWERTE
180:   C000           TH        =    CIA+7
180:   C000           TAST      =    $EA87    ;TASTATURROUT.
190:   C000           DDRB      =    $DD03
200:   C000           ICR       =    CIA+13   ;INTERR.CONTROLREG.
200:   C000           CRB       =    CIA+15
200:   C000           TSTART    =    *+$0500
210:   C000 EA                  NOP
210:   C001 A9 00               LDA  #0
210:   C003 8D 87 C4            STA  DISON
210:   C006 F0 05               BEQ  GO
220:   C008 A9 FF               LDA  #255
220:   C00A 8D 87 C4            STA  DISON
230:   C00D 78        GO        SEI
230:   C00E A9 34               LDA  #<TRACE
230:   C010 8D 14 03            STA  IRQVEK   ;IRQ VEKTOR
240:   C013 A9 C0               LDA  #>TRACE
240:   C015 8D 15 03            STA  IRQVEK+1 ;AUF TRACE
250:   C018 A9 82               LDA  #130
250:   C01A 8D 0D DC            STA  ICR      ;TIMER B INTERRUPT ERLAUBT
250:   C01D A2 FF               LDX  #255
250:   C01F 9A                  TXS           ;STACK INITIAL.
260:   C020 A9 A4               LDA  #>READY  ;STACK FUER TRACE
260:   C022 48                  PHA           ;VORBEREITEN
260:   C023 A9 73               LDA  #<READY
260:   C025 48                  PHA
270:   C026 A9 C0               LDA  #$C0     ;PC=$C000
270:   C028 48                  PHA
270:   C029 A9 00               LDA  #0
270:   C02B 48                  PHA
280:   C02C A9 20               LDA  #32
280:   C02E 48                  PHA
280:   C02F A9 00               LDA  #0
280:   C031 48                  PHA
280:   C032 48                  PHA
280:   C033 48                  PHA
290:   C034 D8        TRACE     CLD
290:   C035 A0 00               LDY  #0
290:   C037 B9 41 C4  TITEL     LDA  TEXT,Y   ;KOPFZEILE
300:   C03A 20 2C C1            JSR  PRIZEI   ;AUSGEBEN
300:   C03D C0 28               CPY  #40
300:   C03F D0 F6               BNE  TITEL
310:   C041 BA                  TSX
310:   C042 BD 06 01            LDA  STACK+6,X ;PC AUSGEBEN
310:   C045 8D 7C C4            STA  TMP+3
310:   C048 20 0E C1            JSR  PRBYTE
320:   C04B BD 05 01            LDA  STACK+5,X
320:   C04E 8D 7D C4            STA  TMP+4
320:   C051 20 0E C1            JSR  PRBYTE
330:   C054 AD 87 C4            LDA  DISON
330:   C057 30 03               BMI  DISOFF
340:   C059 20 74 C2            JSR  DISAS
340:   C05C 20 2A C1  DISOFF    JSR  SPACE
340:   C05F 8A                  TXA           ;SP AUSGEBEN
350:   C060 20 0E C1            JSR  PRBYTE
360:   C063 20 2A C1  REGISTER  JSR  SPACE
360:   C066 BD 01 01            LDA  STACK+1,X
360:   C069 20 0E C1            JSR  PRBYTE
370:   C06C E8                  INX
370:   C06D C0 38               CPY  #56
370:   C06F 90 F2               BCC  REGISTER
370:   C071 20 2A C1            JSR  SPACE
380:   C074 BD 01 01            LDA  STACK+1,X ;PROZESSORSTATUS-FLAGS ANZEIGEN
390:   C077 A2 08               LDX  #8
400:   C079 0A        CONT1     ASL
400:   C07A 48                  PHA
400:   C07B A9 5E               LDA  #94      ;PFEIL
400:   C07D B0 02               BCS  FLSETZ
410:   C07F A9 2D               LDA  #45      ;MINUS
410:   C081 20 2C C1  FLSETZ    JSR  PRIZEI
410:   C084 68                  PLA
420:   C085 CA                  DEX
420:   C086 D0 F1               BNE  CONT1
420:   C088 20 2A C1            JSR  SPACE
420:   C08B A2 08               LDX  #8       ;USERPORT ANZEIGEN
430:   C08D AD 01 DD            LDA  DDRB-2
440:   C090 0A        CONT2     ASL
440:   C091 48                  PHA
440:   C092 A9 31               LDA  #$31
440:   C094 B0 02               BCS  USETZ
450:   C096 A9 30               LDA  #$30
450:   C098 20 2C C1  USETZ     JSR  PRIZEI
450:   C09B 68                  PLA
460:   C09C CA                  DEX
460:   C09D D0 F1               BNE  CONT2
460:   C09F A2 08               LDX  #8       ;DATENRICHTUNGSREGISTER
460:   C0A1 AD 03 DD            LDA  DDRB     ;LADEN
470:   C0A4 1E 41 04  CONT3     ASL  SCREEN+65,X
470:   C0A7 4A                  LSR           ;ENTSPRECHENDE BITS DER ANZEIGE
480:   C0A8 7E 41 04            ROR  SCREEN+65,X ;WERDEN INVERTIERT
480:   C0AB CA                  DEX
480:   C0AC D0 F6               BNE  CONT3
```

```
490:   C0AE 20 38 C1 WAIT     JSR  GET      ;WARTET AUF
490:   C0B1 C9 17             CMP  #23      ;F1,F3,F5,F7 ODER X
490:   C0B3 F0 3C             BEQ  EXIT
500:   C0B5 C9 05             CMP  #5
500:   C0B7 F0 16             BEQ  FAST     ;UND VERZWEIGT ENTSPRECHEND
510:   C0B9 C9 06             CMP  #6
510:   C0BB F0 2A             BEQ  SLOW
520:   C0BD C9 03             CMP  #3
520:   C0BF F0 07             BEQ  WAITL
520:   C0C1 C9 04             CMP  #4
520:   C0C3 D0 E9             BNE  WAIT
530:   C0C5 4C 3E C1          JMP  SET
540:   C0C8 20 38 C1 WAITL    JSR  GET      ;WARTET BIS F7 WIEDER LOSGELASSEN
540:   C0CB C9 40             CMP  #64
540:   C0CD D0 F9             BNE  WAITL
550:   C0CF A9 16    FAST     LDA  #$16
550:   C0D1 8D 06 DC          STA  TL
560:   C0D4 A9 00             LDA  #0       ;TIMER LADEN
560:   C0D6 8D 07 DC          STA  TH
560:   C0D9 A9 11             LDA  #17
560:   C0DB 8D 0F DC          STA  CRB      ;UND STARTEN
560:   C0DE AD 0D DC          LDA  $DC0D    ;ENDE IRQ
570:   C0E1 68                PLA
570:   C0E2 A8                TAY
570:   C0E3 68                PLA
570:   C0E4 AA                TAX
580:   C0E5 68                PLA
580:   C0E6 40                RTI
590:   C0E7 A2 00    SLOW     LDX  #0
590:   C0E9 CA       CONT4    DEX
590:   C0EA D0 FD             BNE  CONT4
590:   C0EC 88                DEY
590:   C0ED D0 FA             BNE  CONT4
600:   C0EF F0 DE             BEQ  FAST
610:   C0F1 A9 02    EXIT     LDA  #2       ;X TASTE
610:   C0F3 8D 0D DC          STA  ICR
610:   C0F6 A9 31             LDA  #$31
610:   C0F8 8D 14 03          STA  IRQVEK   ;SETZT IRQ VEKTOR AUF
620:   C0FB A9 EA             LDA  #$EA     ;ALTEN WERT
620:   C0FD 8D 15 03          IRQVEK+1
620:   C100 A9 04             LDA  #4
620:   C102 8D 8B 02          STA  $028B
620:   C105 A9 10             LDA  #16      ;REPEAT KORRIG.
630:   C107 8D 8C 02          STA  $028C
630:   C10A 58                CLI
630:   C10B 4C 74 A4          JMP  READY+1 ;SPRUNG ZU BASIC
640:   C10E 48       PRBYTE   PHA           ;1 BYTE ALS 2 HEXZ. AUF SCHIRM
640:   C10F 8E 79 C4          STX  TMP
640:   C112 4A                LSR
640:   C113 4A                LSR
640:   C114 4A                LSR
640:   C115 4A                LSR
650:   C116 AA                TAX
650:   C117 BD 69 C4          LDA  HEX,X
650:   C11A 20 2C C1          JSR  PRIZEI
660:   C11D 68                PLA
660:   C11E 29 0F             AND  #$0F
660:   C120 AA                TAX
660:   C121 BD 69 C4          LDA  HEX,X
660:   C124 AE 79 C4          LDX  TMP
670:   C127 4C 2C C1          JMP  PRIZEI
680:   C12A A9 20    SPACE    LDA  #$20
690:   C12C 29 3F    PRIZEI   AND  #$3F     ;WEGEN BILDSCHIRMCODE
700:   C12E 99 00 04          STA  SCREEN,Y
700:   C131 A9 01             LDA  #1       ;FARBE SETZEN
700:   C133 99 00 D8          STA  COL,Y
700:   C136 C8                INY
700:   C137 60                RTS
710:   C138 20 87 EA GET      JSR  TAST     ;HOLT TASTENCODE
710:   C13B A5 CB             LDA  $CB
710:   C13D 60                RTS
720:   C13E 20 74 C2 SET      JSR  DISAS
720:   C141 A0 28             LDY  #40      ;F1 TASTE
730:   C143 B9 00 04 START    LDA  SCREEN,Y
730:   C146 09 80             ORA  #128
730:   C148 99 00 04          STA  SCREEN,Y
730:   C14B 8C 79 C4          STY  TMP
740:   C14E 20 82 C1          JSR  ZEICHHOL ;EINGABEROUT.
740:   C151 AC 79 C4          LDY  TMP
740:   C154 C9 0D             CMP  #13
740:   C156 D0 11             BNE  LINKS
750:   C158 C0 64             CPY  #100
750:   C15A 10 03             BPL  MREAD
750:   C15C 4C A2 C1          JMP  READ
750:   C15F 60       MREAD    RTS           ;ZUR LESEROUTINE
760:   C160 B9 00 04 REVERSE  LDA  SCREEN,Y ;INVERTIEREN
770:   C163 29 7F             AND  #127
770:   C165 99 00 04          STA  SCREEN,Y
770:   C168 60                RTS
780:   C169 C9 11    LINKS    CMP  #17      ;CURSOR LINKS
780:   C16B D0 06             BNE  RECHTS
780:   C16D 20 60 C1          JSR  REVERSE
780:   C170 88                DEY
780:   C171 D0 D0             BNE  START
790:   C173 C9 1D    RECHTS   CMP  #29      ;CURSOR RECHTS
790:   C175 D0 06             BNE  CONT6
790:   C177 20 60 C1          JSR  REVERSE
790:   C17A C8                INY
790:   C17B D0 C6             BNE  START
800:   C17D 20 2C C1 CONT6    JSR  PRIZEI   ;WERT ANZEIGEN
800:   C180 D0 C1             BNE  START
800:   C182 A9 00    ZEICHHOL LDA  #0       ;TAST.ABFRAGE
810:   C184 85 C6             STA  $C6
810:   C186 20 87 EA          JSR  TAST
810:   C189 AD 77 02          LDA  $277
820:   C18C C9 00             CMP  #0
820:   C18E F0 F2             BEQ  ZEICHHOL
820:   C190 A0 00             LDY  #0
820:   C192 8C 77 02          STY  $277     ;TASTATURPUFFERSTART
830:   C195 C9 85             CMP  #$85     ;F1 TASTE
830:   C197 F0 E9             BEQ  ZEICHHOL
840:   C199 A0 40             LDY  #64      ;VERZOEGERUNG REPEATZAEHLER
840:   C19B 8C 8B 02          STY  $28B
840:   C19E 8C 8C 02          STY  $28C
840:   C1A1 60                RTS
850:   C1A2 20 60 C1 READ     JSR  REVERSE
850:   C1A5 A0 28             LDY  #40      ;PC LESEN
850:   C1A7 BA                TSX
850:   C1A8 20 F6 C1          JSR  BYT
850:   C1AB 9D 06 01          STA  STACK+6,X
860:   C1AE 8D 7C C4          STA  TMP+3
860:   C1B1 C8                INY
860:   C1B2 20 F6 C1          JSR  BYT
870:   C1B5 9D 05 01          STA  STACK+5,X
870:   C1B8 8D 7D C4          STA  TMP+4
870:   C1BB C8                INY
870:   C1BC C8                INY                ;SP UEBERLESEN
870:   C1BD C8                INY
870:   C1BE C8                INY
880:   C1BF C8       LOOP     INY                ;REGISTER LESEN
880:   C1C0 20 F6 C1          JSR  BYT
880:   C1C3 9D 01 01          STA  STACK+1,X
880:   C1C6 C8                INY
880:   C1C7 E8                INX
890:   C1C8 C0 38             CPY  #56
890:   C1CA D0 F3             BNE  LOOP
900:   C1CC C8                INY
900:   C1CD A9 00             LDA  #0
900:   C1CF 8D 79 C4          STA  TMP
900:   C1D2 B9 00 04 LOOP1    LDA  SCREEN,Y ;FLAGS LESEN
900:   C1D5 29 1F             AND  #%00011111
910:   C1D7 C9 1E             CMP  #30
910:   C1D9 2E 79 C4          ROL  TMP
920:   C1DC C8                INY
920:   C1DD C0 41             CPY  #65
920:   C1DF D0 F1             BNE  LOOP1
920:   C1E1 BA                TSX
920:   C1E2 AD 79 C4          LDA  TMP
920:   C1E5 29 FB             AND  #%11111011 ;I FLAG LOESCHEN
930:   C1E7 9D 04 01          STA  STACK+4,X
930:   C1EA 20 74 C2          JSR  DISAS
930:   C1ED 20 D5 C3          JSR  READPC
930:   C1F0 20 74 C2          JSR  DISAS
930:   C1F3 4C AE C0          JMP  WAIT
940:   C1F6 20 09 C2 BYT      JSR  LBIT     ;8 BIT ZAHL VON SCREEN HOLEN
940:   C1F9 0A                ASL
940:   C1FA 0A                ASL
940:   C1FB 0A                ASL
940:   C1FC 0A                ASL
940:   C1FD C8                INY
940:   C1FE 8D 79 C4          STA  TMP
950:   C201 20 09 C2          JSR  LBIT
950:   C204 18                CLC
950:   C205 6D 79 C4          ADC  TMP
950:   C208 60                RTS
960:   C209 B9 00 04 LBIT     LDA  SCREEN,Y ;1 BYTE VON SCREEN IN HEX
960:   C20C C9 30             CMP  #48
960:   C20E B0 03             BCS  ZAHL
960:   C210 69 39             ADC  #57
970:   C212 38                SEC
970:   C213 E9 30    ZAHL     SBC  #48
970:   C215 60                RTS
                     ;BERECHNET ABS. ADRESSEN BEI BRANCHES
990:   C216 AD 84 C4 OFFSET   LDA  COD2
990:   C219 30 18             BMI  BMINUS
1000:  C21B 18                CLC                ;BRANCH VOR
1000:  C21C 69 02             ADC  #2
1000:  C21E 6D 7D C4          ADC  TMP+4
1000:  C221 8D 80 C4          STA  TMP1
1000:  C224 AD 7C C4          LDA  TMP+3
1010:  C227 69 00             ADC  #0
1010:  C229 20 0E C1          JSR  PRBYTE
1010:  C22C AD 80 C4          LDA  TMP1
1010:  C22F 20 0E C1          JSR  PRBYTE
1010:  C232 60                RTS
                     ;BRANCH RUECK
1020:  C233 49 FF    BMINUS   EOR  #255
1020:  C235 38                SEC
1020:  C236 E9 01             SBC  #1
1020:  C238 8D 79 C4          STA  TMP
1030:  C23B AD 7D C4          LDA  TMP+4
1030:  C23E ED 79 C4          SBC  TMP
1030:  C241 8D 80 C4          STA  TMP1
1030:  C244 AD 7C C4          LDA  TMP+3
1040:  C247 E9 00             SBC  #0
1040:  C249 20 0E C1          JSR  PRBYTE
1040:  C24C AD 80 C4          LDA  TMP1
1040:  C24F 20 0E C1          JSR  PRBYTE
1040:  C252 60                RTS                ;ENDE OFFSET
                     ;LAENGE OPCODE
                     ;BERECHNET BEFEHLSLAENGE (1,2 OD. 3 BYTES)
                     ;OPCODE IN AKKU UEBERGEBEN
                     ;IM Y-REG. STEHT DANN BEFEHLSLAENGE
1090:  C253 A0 01    LAENGE   LDY  #1
1100:  C255 C9 20             CMP  #$20
1100:  C257 F0 18             BEQ  LO01
1100:  C259 29 9F             AND  #$9F
1100:  C25B F0 16             BEQ  LO03
1110:  C25D 29 1F             AND  #$1F
1110:  C25F C9 09             CMP  #9
1110:  C261 F0 0F             BEQ  LO02
1110:  C263 C9 19             CMP  #$19
1120:  C265 F0 0A             BEQ  LO01
1120:  C267 29 0D             AND  #$0D
1120:  C269 C9 08             CMP  #8
1120:  C26B F0 06             BEQ  LO03
1130:  C26D 29 08             AND  #8
1130:  C26F F0 01             BEQ  LO02
1140:  C271 C8       LO01     INY
1140:  C272 C8       LO02     INY
1140:  C273 60       LO03     RTS
1150:  C274 8C 81 C4 DISAS    STY  OLDY
1150:  C277 8E 82 C4          STX  OLDX     ;REGISTER RETTEN
1160:  C27A A2 14             LDX  #20
1160:  C27C A9 20             LDA  #" "
1160:  C27E 9D 78 04 CLE1     STA  SCREEN+120,X ;ZEILE FUER DISAS. LOESCHEN
1170:  C281 CA                DEX
1170:  C282 D0 FA             BNE  CLE1
                     ;PC L/H IN TMP+4 / TMP+3
                     ;BELEGT 'RAM' POINTER ZUM LESEN
1190:  C284 AD 7D C4          LDA  TMP+4
1190:  C287 8D BF C2          STA  PPOIN1+1
1190:  C28A 8D 7F C4          STA  TMP+6
1190:  C28D AD 7C C4          LDA  TMP+3
1190:  C290 8D C0 C2          STA  PPOIN1+2
1190:  C293 8D 7E C4          STA  TMP+5
```

Listing von »Trace und Single Step« (Fortsetzung)

```
1200:  C296 EE 7F C4           INC  TMP+6
1200:  C299 AD 7F C4           LDA  TMP+6
1200:  C29C 8D C5 C2           STA  PPOIN2+1
1200:  C29F D0 03              BNE  L005
1210:  C2A1 EE 7E C4           INC  TMP+5
1210:  C2A4 AD 7E C4 L005      LDA  TMP+5
1210:  C2A7 8D C6 C2           STA  PPOIN2+2
1210:  C2AA EE 7F C4           INC  TMP+6
1220:  C2AD AD 7F C4           LDA  TMP+6
1220:  C2B0 8D CB C2           STA  PPOIN3+1
1220:  C2B3 D0 03              BNE  L006
1230:  C2B5 EE 7E C4           INC  TMP+5
1230:  C2B8 AD 7E C4 L006      LDA  TMP+5
1230:  C2BB 8D CC C2           STA  PPOIN3+2
1240:  C2BE AD 00 C0 PPOIN1    LDA  $C000
1240:  C2C1 8D 83 C4           STA  COD1     ;LIEST OPCODES
1250:  C2C4 AD 00 C0 PPOIN2    LDA  $C000
1250:  C2C7 8D 84 C4           STA  COD2
1260:  C2CA AD 00 C0 PPOIN3    LDA  $C000
1260:  C2CD 8D 85 C4           STA  COD3
1270:  C2D0 AD 83 C4           LDA  COD1
1270:  C2D3 20 53 C2           JSR  LAENGE
1270:  C2D6 98                 TYA
1270:  C2D7 8D 86 C4           STA  CODEL
1270:  C2DA AA                 TAX
1270:  C2DB A0 78              LDY  #120     ;BYTES AB PC AUSGEBEN
1280:  C2DD AD 83 C4           LDA  COD1
1280:  C2E0 20 0E C1           JSR  PRBYTE
1280:  C2E3 20 2A C1           JSR  SPACE
1280:  C2E6 CA                 DEX
1290:  C2E7 F0 12              BEQ  L007
1290:  C2E9 AD 84 C4           LDA  COD2
1290:  C2EC 20 0E C1           JSR  PRBYTE
1290:  C2EF 20 2A C1           JSR  SPACE
1300:  C2F2 CA                 DEX
1300:  C2F3 F0 06              BEQ  L007
1300:  C2F5 AD 85 C4           LDA  COD3
1300:  C2F8 20 0E C1           JSR  PRBYTE
1310:  C2FB A0 81    L007      LDY  #129
1310:  C2FD AD 83 C4           LDA  COD1     ;GIBT OPCODE (3 BUCHSTABEN) AUS
1320:  C300 18                 CLC
1320:  C301 A9 C5              LDA  #>TSTART
1320:  C303 8D 79 C4           STA  TMP
1320:  C306 AD 83 C4           LDA  COD1
1320:  C309 6D 83 C4           ADC  COD1
1320:  C30C 90 03              BCC  L008
1330:  C30E EE 79 C4           INC  TMP
1330:  C311 18       L008      CLC
1330:  C312 6D 83 C4           ADC  COD1
1330:  C315 90 03              BCC  L009
1330:  C317 EE 79 C4           INC  TMP
1330:  C31A 8D 26 C3 L009      STA  OPP+1    ;POINTER BELEGEN
1340:  C31D AD 79 C4           LDA  TMP
1340:  C320 8D 27 C3           STA  OPP+2
1340:  C323 A2 00              LDX  #0
1340:  C325 BD 00 C0 OPP       LDA  $C000,X
1340:  C328 20 2C C1           JSR  PRIZEI
1340:  C32B E8                 INX
1350:  C32C E0 03              CPX  #3
1350:  C32E D0 F5              BNE  OPP
1350:  C330 20 2A C1           JSR  SPACE
1360:  C333 AD 86 C4           LDA  CODEL
1360:  C336 C9 02              CMP  #2
1360:  C338 10 03              BPL  CODE2
1360:  C33A 4C CE C3           JMP  ENDAS
1370:  C33D AE 83 C4 CODE2     LDX  COD1
1370:  C340 BD 00 C8           LDA  TADR,X  ;ADRESS. FESTSTELLEN
1370:  C343 AA                 TAX
                     ; '#$','$' OD.'($' AUSGEBEN
1390:  C344 E0 06              CPX  #6
1390:  C346 30 05              BMI  L0010
1390:  C348 A9 28              LDA  #"("
1390:  C34A 20 2C C1           JSR  PRIZEI
1400:  C34D E0 02    L0010     CPX  #2
1400:  C34F D0 05              BNE  L0011
1400:  C351 A9 23              LDA  #"#"
1410:  C353 20 2C C1           JSR  PRIZEI
1410:  C356 A9 24    L0011     LDA  #"$"
1410:  C358 20 2C C1           JSR  PRIZEI
1420:  C35B E0 01              CPX  #1
1420:  C35D 10 06              BPL  L0012
1420:  C35F 20 16 C2           JSR  OFFSET
1420:  C362 4C CE C3           JMP  ENDAS
1430:  C365 AD 86 C4 L0012     LDA  CODEL    ;OPERANDEN AUSGEBEN
1430:  C368 C9 03              CMP  #3
1430:  C36A F0 09              BEQ  CODL3
1440:  C36C AD 84 C4           LDA  COD2
1440:  C36F 20 0E C1           JSR  PRBYTE
1440:  C372 4C 81 C3           JMP  L0013
1450:  C375 AD 85 C4 CODL3     LDA  COD3
1450:  C378 20 0E C1           JSR  PRBYTE
1450:  C37B AD 84 C4           LDA  COD2
1450:  C37E 20 0E C1           JSR  PRBYTE
1460:  C381 E0 04    L0013     CPX  #4
1460:  C383 30 49              BMI  ENDAS
                     ; ',X',',Y',',X)','),Y' OD. ')' AUSGEBEN
1480:  C385 D0 0D              BNE  L0014
1480:  C387 A9 2C              LDA  #","
1480:  C389 20 2C C1           JSR  PRIZEI
1490:  C38C A9 58              LDA  #"X"
1490:  C38E 20 2C C1           JSR  PRIZEI
1490:  C391 4C CE C3           JMP  ENDAS
1500:  C394 E0 05    L0014     CPX  #5
1500:  C396 D0 0D              DNE  L0015
1510:  C398 A9 2C              LDA  #","
1510:  C39A 20 2C C1           JSR  PRIZEI
1510:  C39D A9 59              LDA  #"Y"
1510:  C39F 20 2C C1           JSR  PRIZEI
1510:  C3A2 4C CE C3           JMP  ENDAS
1520:  C3A5 E0 06    L0015     CPX  #6
1520:  C3A7 D0 12              BNE  L0016
1520:  C3A9 A9 2C              LDA  #","
1520:  C3AB 20 2C C1           JSR  PRIZEI
1530:  C3AE A9 58              LDA  #"X"
1530:  C3B0 20 2C C1           JSR  PRIZEI
1530:  C3B3 A9 29              LDA  #")"
1530:  C3B5 20 2C C1           JSR  PRIZEI
1530:  C3B8 4C CE C3           JMP  ENDAS
```

```
1540:  C3BB A9 29    L0016     LDA  #")"
1540:  C3BD 20 2C C1           JSR  PRIZEI
1550:  C3C0 E0 08              CPX  #8
1550:  C3C2 F0 0A              BEQ  ENDAS
1550:  C3C4 A9 2C              LDA  #","
1550:  C3C6 20 2C C1           JSR  PRIZEI
1560:  C3C9 A9 59              LDA  #"Y"
1560:  C3CB 20 2C C1           JSR  PRIZEI
1570:  C3CE AC 81 C4 ENDAS     LDY  OLDY     ;REG. HOLEN
1570:  C3D1 AE 82 C4           LDX  OLDX
1570:  C3D4 60                 RTS           ;ENDE DISAS
1580:  C3D5 A0 78    READPC    LDY  #120
1580:  C3D7 20 43 C1           JSR  START    ;DIS.ZEILE EDIT.
1580:  C3DA 20 60 C1           JSR  REVERSE
1580:  C3DD A0 78              LDY  #120
                     ;LIEST BYTES UND SPEICHERT IN RAM
1600:  C3DF 20 F6 C1           JSR  BYT
1600:  C3E2 8D 83 C4           STA  COD1
1600:  C3E5 C8                 INY
1600:  C3E6 C8                 INY
1600:  C3E7 20 F6 C1           JSR  BYT
1610:  C3EA 8D 84 C4           STA  COD2
1610:  C3ED C8                 INY
1610:  C3EE C8                 INY
1610:  C3EF 20 F6 C1           JSR  BYT
1610:  C3F2 8D 85 C4           STA  COD3
1620:  C3F5 AD 7D C4           LDA  TMP+4
1620:  C3F8 8D 05 C4           STA  PREAD1+1
1630:  C3FB AD 7C C4           LDA  TMP+3
1630:  C3FE 8D 06 C4           STA  PREAD1+2
1630:  C401 AD 83 C4           LDA  COD1
1630:  C404 8D 00 C0 PREAD1    STA  $C000
1640:  C407 AC 86 C4           LDY  CODEL
1640:  C40A C0 02              CPY  #2
1640:  C40C 30 32              BMI  EREAD
1650:  C40E AD 7D C4           LDA  TMP+4
1650:  C411 18                 CLC
1650:  C412 69 01              ADC  #1
1650:  C414 8D 23 C4           STA  PREAD2+1
1660:  C417 AD 7C C4           LDA  TMP+3
1660:  C41A 69 00              ADC  #0
1660:  C41C 8D 24 C4           STA  PREAD2+2
1670:  C41F AD 84 C4           LDA  COD2
1670:  C422 8D 00 C0 PREAD2    STA  $C000
1680:  C425 C0 03              CPY  #3
1680:  C427 30 17              BMI  EREAD
1690:  C429 AD 7D C4           LDA  TMP+4
1690:  C42C 18                 CLC
1690:  C42D 69 02              ADC  #2
1690:  C42F 8D 3E C4           STA  PREAD3+1
1700:  C432 AD 7C C4           LDA  TMP+3
1700:  C435 69 00              ADC  #0
1700:  C437 8D 3F C4           STA  PREAD3+2
1710:  C43A AD 85 C4           LDA  COD3
1710:  C43D 8D 00 C0 PREAD3    STA  $C000
1720:  C440 60       EREAD     RTS           ;ALLE BYTES GELESEN. ENDE READPC
1730:  C441 20 50 43 TEXT      .ASC " PC  SP YR XR AC NV-BDIZC 76543210     "
1740:  C469 30 31 32 HEX       .ASC "0123456789ABCDEF"
1750:  C479 00 00 00 TMP       .BYT 0,0,0,0,0,0,0 ;ARBEITSVARIABLEN
1760:  C480 00       TMP1      .BYT 0
1770:  C481 00       OLDY      .BYT 0
1780:  C482 00       OLDX      .BYT 0
1790:  C483 00       COD1      .BYT 0
1790:  C484 00       COD2      .BYT 0
1790:  C485 00       COD3      .BYT 0
1800:  C486 00       CODEL     .BYT 0
1800:  C487 00       DISON     .BYT 0
1810:  C500                    *=   TSTART
                     ;TAB. OPCODES (L-BYTE TSTART MUSS 0 SEIN)
1820:  C500 42 52 4B           .ASC "BRKORA?????????ORAASL???PHPORAASL"
1830:  C521 3F 3F 3F           .ASC "??????ORAASL???BPLORA??????"
1840:  C53C 3F 3F 3F           .ASC "???ORAASL???CLCORA?????????"
1850:  C557 4F 52 41           .ASC "ORAASL???JSRAND??????BITAND"
1860:  C572 52 4F 4C           .ASC "ROL???PLPANDROL???BITANDROL"
1870:  C58D 3F 3F 3F           .ASC "???BMIAND?????????ANDROL???"
1880:  C5A8 53 45 43           .ASC "SECAND?????????ANDROL???RTI"
1890:  C5C3 45 4F 52           .ASC "EOR?????????EORLSR???PHAEOR"
1900:  C5DE 4C 53 52           .ASC "LSR???JMPEORLSR???BVCEOR???"
1910:  C5F9 3F 3F 3F           .ASC "??????EORLSR???CLIEOR??????"
1920:  C614 3F 3F 3F           .ASC "???EORLSR???RTSADC?????????"
1930:  C62F 41 44 43           .ASC "ADCROR???PLAADCROR???JMPADC"
1940:  C64A 52 4F 52           .ASC "ROR???BVSADC?????????ADCROR"
1950:  C665 3F 3F 3F           .ASC "???SEIADC?????????ADCROR???"
1960:  C680 3F 3F 3F           .ASC "???STA??????STYSTASTX???DEY"
1970:  C69B 3F 3F 3F           .ASC "???TXA???STYSTASTX???BCCSTA"
1980:  C6B6 3F 3F 3F           .ASC "??????STYSTASTX???TYASTATXS"
1990:  C6D1 3F 3F 3F           .ASC "??????STA??????LDYLDALDX???"
2000:  C6EC 4C 44 59           .ASC "LDYLDALDX???TAYLDATAX???LDY"
2010:  C707 4C 44 41           .ASC "LDALDX???BCSLDA??????LDYLDA"
2020:  C722 4C 44 58           .ASC "LDX???CLVLDATSX???LDYLDALDX"
2030:  C73D 3F 3F 3F           .ASC "???CPYCMP??????CPYCMPDEC???"
2040:  C758 49 4E 59           .ASC "INYCMPDEX???CPYCMPDEC???BNE"
2050:  C773 43 4D 50           .ASC "CMP?????????CMPDEC???CLDCMP"
2060:  C78E 3F 3F 3F           .ASC "?????????CMPDEC???CPXSBC???"
2070:  C7A9 3F 3F 3F           .ASC "???CPXSBCINC???INXSBCNOP???"
2080:  C7C4 43 50 58           .ASC "CPXSBCINC???BEQSBC?????????"
2090:  C7DF 53 42 43           .ASC "SBCINC???SEDSBC?????????SBC"
2100:  C7FA 49 4E 43           .ASC "INC???"
                     ;TABELLE ADRESSIERUNG (KEINE STANDARDWERTE NUR FUER DISAS)
2120:  C800 01 06 01 TADR      .BYT 1,6,1,1,1,3,3,1,1,2,1,1
2130:  C80C 01 03 03           .BYT 1,3,3,1,0,7,1,1,1,4,4,1,1,5
2140:  C81A 01 01 01           .BYT 1,1,1,4,4,1,3,6,1,1,3,3,3,1
2150:  C828 01 02 01           .BYT 1,2,1,1,3,3,3,1,0,7,1,1,1,4
2160:  C836 04 01 01           .BYT 4,1,1,5,1,1,1,4,4,1,1,6,1,1
2170:  C844 01 03 03           .BYT 1,3,3,1,1,2,1,1,3,3,3,1,0,7
2180:  C852 01 01 01           .BYT 1,1,1,4,4,1,1,5,1,1,1,4,4,1
2190:  C860 01 06 01           .BYT 1,6,1,1,1,3,3,1,1,2,1,1,8,3
2200:  C86E 03 01 00           .BYT 3,1,0,7,1,1,1,4,4,1,1,5,1,1,1
2210:  C87D 04 04 01           .BYT 4,4,1,1,6,1,1,3,3,3,1,1,1,1
2220:  C88B 01 03 03           .BYT 1,3,3,3,1,0,7,1,1,4,4,5,1,1
2230:  C899 05 01 01           .BYT 5,1,1,1,4,1,1,2,6,2,1,3,3,3
2240:  C8A7 01 01 02           .BYT 1,1,2,1,1,3,3,3,1,0,7,1,1,4
2250:  C8B5 04 05 01           .BYT 4,5,1,1,5,1,1,4,4,5,1,2,6,1
2260:  C8C3 01 03 03           .BYT 1,3,3,3,1,1,2,1,1,3,3,3,1,0
2270:  C8D1 07 01 01           .BYT 7,1,1,1,4,4,1,1,5,1,1,1,4,4
2280:  C8DF 01 02 06           .BYT 1,2,6,1,1,3,3,3,1,1,2,1,1,3
2290:  C8ED 03 03 01           .BYT 3,3,1,0,7,1,1,1,4,4,1,1,5,1
2300:  C8FB 01 01 04           .BYT 1,1,4,4,1
```

Listing von »Trace und Single Step« (Schluß)

Maschinenprogramme auf Tastendruck

Mit einer kleinen Routine kann man ein Maschinenprogramm mit einem Tastendruck aufrufen. Dafür benutzt man ein Zeichen, das normalerweise nicht oder nur selten verwendet wird. Ich habe mich für das @-Zeichen entschieden.

Im Interpreter existiert eine Schleife, die einen Basic-Befehl holt und ausführt.

```
A7E1   JMP   (0308)  ; zeigt normalerweise auf A7E4
A7E4   JSR   0073    ; nächstes Zeichen aus Basic-Text holen
A7E7   JSR   A7ED    ; Statement ausführen
A7EA   JMP   A7AE    ; zurück zur Interpreterschleife
```

In den Speicherzellen 0308 und 0309 (776 und 777 dez) liegt ein Zeiger, der normalerweise auf den Beginn dieser Schleife zeigt. Verbiegt man nun den Zeiger auf eine eigene Routine, kann man den Basic-Befehl auf das eigene Zeichen überprüfen.

Wird es erkannt, springt man auf den Anfang des gewünschten Unterprogramms. Wurde das Zeichen nicht vorgefunden, macht man in der Schleife normal weiter.

Dieses Verfahren verwende ich bei der Programmierhilfe »Merge« aus Ausgabe 4/84, die normalerweise mit SYS 50000 gestartet werden muß. Es kann aber auch für andere Maschinenprogramme umgeschrieben werden.

»Merge« belegt den Speicherbereich 50000 bis 50264. Die eigene Routine beginnt auf Adresse 49152 (C000 hex). Der Wert in den Adressen 0308 und 0309 muß deshalb auf C000 abgeändert werden. Der Computer durchläuft dann vor jedem Befehl, den er ausführen soll, folgende Schleife:

```
C000   JSR   0073   ; nächstes Zeichen holen
C003   CMP     40   ; Vergleich mit @-Zeichen
C005   BEQ          ; verzweigen wenn erkannt
C007   JSR   0079   ; Flags setzen
C00A   JMP   A7E7   ; Rücksprung
C00D   JSR   0073   ; nächstes Zeichen holen
C010   JSR   C350   ; zur eigenen Routine
C013   JMP   A7AE   ; Rücksprung
```

Nach dem Drücken von @ und RETURN wird nun das Programm ab Adresse 50000 (C350 Hex) ausgeführt. Auf die anderen Befehle hat diese Routine keinen Einfluß. Eine Hürde gibt es noch zu meistern. Die Änderung der Adressen 0308 und 0309 ist auf der Basic-Ebene mit POKE nicht möglich. Dies ist auch verständlich, da POKE auch ein Basic-Befehl ist und durch die Änderung der ersten Adresse der Einsprung verändert wird.

Deshalb muß diese Adreßänderung in Maschinensprache durchgeführt werden.

```
C100   LDA     00   ; Lade Akku mit 00
C102   STA   0308   ; Speichere Akku nach 0308
C105   LDA     C0   ; Lade Akku mit C0
C107   STA   0309   ; Speichere Akku nach 0309
C10A   RTS          ; Rückkehr nach Basic
```

Basic-Lader für Befehlserweiterung

```
240     FOR I= 49152 TO 49152 + 21
250     READ Q : POKE I, Q
260     NEXT
300     FOR I = 49408 TO 49408 + 10
310     READ Q : POKE I, Q
320     NEXT : SYS 49408
11000   DATA 32,115,0,201,64,240,6,32,121
12000   DATA 0,76,231,167,32,115,0,32,80
13000   DATA 195,76,174,167
14000   DATA 169,0,141,8,3,169,192,141,9,3,96
```

Diesen Basic-Lader tippt man hinter das Programm »Merge« und speichert es gemeinsam ab.

Die Zeilennummern sind so gewählt, daß man sie direkt zum Basic-Lader von »Merge« dazutippen kann. In Zeile 10260 müssen aber dann die letzten fünf Nullen gelöscht werden.

Mit SYS 49408 wird die Befehlserweiterung aktiviert und steht dann zur Benutzung bereit.

(Patrik Fleig/rg)

Fast Tape — die schnelle Kassette

Dieses Programm für den VC 20 ermöglicht Ihnen einen zehnmal schnelleren Lade- und Abspeichervorgang, das lange Warten bei 16- oder 8-KByte-Programmen hat nun ein Ende.

Das Maschinenprogramm benötigt etwa 750 Byte Ihres Basic-Speichers, was aber bei einer 16-KByte-Erweiterung nicht viel ausmacht. Das Programm läuft auch auf der Grundversion, doch ist es dann nicht so rentabel, weil es zu lange dauert, bis man Fast Tape geladen hat, und außerdem sind die Ladezeiten bei Programmen für die Grundversion noch erträglich. Aufgerufen wird Fast Tape mit »!L« zum Laden eines Programms und mit »!S« zum Abspeichern.

Auf die Ausgabe von »Loading« während des Ladevorgangs sowie »Saving« während des Abspeicherns wurde verzichtet. Ebenfalls muß auf den Befehl »VERIFY« und das schnelle Laden/Abspeichern von Daten-Files verzichtet werden. »Fast Tape« liegt als Basic-Lader vor.

Das Eintippen der DATAs erfordert sehr viel Sorgfalt. Speichern Sie das Programm auf jeden Fall ab, bevor Sie es starten, denn es löscht sich selbständig. Auf gröbere Fehler in den DATAs macht Sie das Programm selbst aufmerksam.

(Thomas Kolbe/ev)

```
1 REM**FAST TAPE**
2 REM*BY THOMAS  *
3 REM*     KOLBE  *
4 REM*************
5 POKE56,PEEK(56)-3:A%=PEEK(56):POKE52,A
```

```
%:POKE51,0:POKE55,0:AS=256*A%:A=0
10 DATA32,115,0,32,209,225,160,0,196,183
,240,8,177,187,153,66,3,200,208
11 DATA244,169,0,153,66,3,200,192,16,144
,246,165,185,141,65,3,165,43,141,61
12 DATA3,165,44,141,62,3,165,45,141,63,3
,165,46,141,64,3,169,255,141,60,3
13 DATA32,77,248,134,172,132,173,169,132
,133,174,169,3,133,175,160,1,132,186
14 DATA32,251,27,165,43,133,172,165,44,1
33,173,165,45,133,174,165,46,133,175
15 DATA160,1,132,186,76,251,27,32,115,0,
32,209,225,160,1,132,186,32,77,248
16 DATA134,172,132,173,169,132,133,174,1
69,3,133,175,32,65,28,173,60,3,201
17 DATA255,208,227,160,99,32,230,241,160
,0,185,66,3,32,210,255,200,192,16
18 DATA208,245,160,0,196,183,240,10,177,
187,217,66,3,208,196,200,208,242,173
19 DATA65,3,201,0,240,21,234,234,162,0,1
89,61,3,149,172,232,224,4,208,246
20 DATA160,1,132,186,76,65,28,165,43,133
,172,165,44,133,173,160,1,132,186
21 DATA173,63,3,56,237,61,3,133,45,173,6
4,3,237,62,3,133,46,24,165,45,101
22 DATA43,133,174,133,45,165,46,101,44,1
33,175,133,46,76,65,28,32,183,248
23 DATA32,160,28,169,2,32,179,28,136,192
,9,208,246,152,32,179,28,162,8,136
24 DATA208,247,132,215,177,172,32,181,28
,162,5,230,172,208,4,230,173,202,202
25 DATA165,172,197,174,165,173,229,175,1
44,231,184,165,215,32,181,28,162,9
26 DATA136,208,246,200,132,192,88,76,8,2
53,32,148,248,32,160,28,132,215,169
27 DATA39,141,40,145,162,1,32,247,28,38,
189,165,189,201,2,208,245,160,9,32
28 DATA231,28,201,2,240,249,196,189,208,
232,32,231,28,136,208,246,145,172
29 DATA69,215,133,215,32,231,28,230,172,
208,2,230,173,165,172,197,174,165
30 DATA173,229,175,165,189,144,229,32,16
4,28,32,58,28,165,189,69,215,240,10
31 DATA165,175,234,201,3,240,3,76,156,22
5,96,201,0,240,12,160,0,132,192,202
32 DATA208,253,136,208,250,120,96,104,10
4,96,162,9,133,189,69,215,133,215
33 DATA169,8,133,163,234,6,189,173,32,14
5,41,247,32,217,28,162,19,184,9,8
34 DATA32,217,28,162,16,198,163,208,232,
96,202,208,253,144,5,162,11,202,208
35 DATA253,141,32,145,96,169,8,133,163,3
2,247,28,38,189,198,163,208,247,165
36 DATA189,96,169,2,44,45,145,240,251,17
3,45,145,142,41,145,44,33,145,10,10
37 DATA10,96,169,22,141,8,3,169,29,141,9
,3,96,32,115,0,240,4,201,33,240,3
38 DATA76,231,199,32,115,0,201,76,240,13
,201,83,208,6
39 DATA32,0,27,76,174,199,76,8,207,32,10
5,27,173,65,3,201,0,240,3,76,174,199
40 DATA169,118,160,195,32,30,203,76,42,1
97,-1
100 READB:IFB>-1THENPOKEAS+A,B:S=S+B:A=A
+1:GOTO100
110 IFS<>73742THENPRINT"DATA ERROR":END
120 POKEAS+81,A%:POKEAS+104,A%:POKEAS+13
2,A%+1:POKEAS+201,A%+1
130 POKEAS+250,A%+1:POKEAS+256,A%+1:POKE
AS+261,A%+1:POKEAS+270,A%+1
140 POKEAS+282,A%+1:POKEAS+308,A%+1:POKE
AS+326,A%+1:POKEAS+338,A%+1
150 POKEAS+351,A%+1:POKEAS+362,A%+1:POKE
AS+374,A%+1:POKEAS+395,A%+1
160 POKEAS+398,A%+1:POKEAS+457,A%+1:POKE
AS+465,A%+1:POKEAS+493,A%+1
170 POKEAS+529,A%+2:POKEAS+559,A%:POKEAS
+568,A%
180 SYS(AS+523)
190 NEW
READY.
```

Listing »Fast Tape« (Basic-Lader)

Zeile	Operation
5	Setzt Basic-Ende um 768 Bytes nach unten
10-40	DATAs der Maschinensprache
100	Einleseroutine
110	Vergleich, ob alle DATAs korrekt eingetippt worden sind
120-170	Anpassung der Maschinensprache an den Adreßbereich
180	FAST TAPE einschalten
190	Programm löschen

Tabelle zum Programmablauf

Master Mind als Vierzeiler

Als Nebenprodukt meiner Einzeilerbemühungen entstand dieses Programm: Bei Master Mind geht es darum, eine Zahl, die sich der Computer »denkt«, zu erraten. Am Anfang gibt man die Stellenzahl der zu erratenden Zahl ein, sie darf maximal acht sein (man hat aber schon mit drei oder vier genug zu knobeln). In der ersten Spalte muß man nun jeweils eine Zahl eingeben, der Computer zeigt in den folgenden drei Spalten an:

1. Anzahl der richtigen Ziffern an der richtigen Stelle
2. Anzahl der richtigen Ziffern an der falschen Stelle
3. Anzahl der Versuche

Beispiel eines Spiels:

```
Stellen? 4
? 1123   1  0  1
? 4456   0  0  2
? 7789   2  1  3
? 8989   1  0  4
? 7979   1  1  5
? 7187   4  0  6
```

Es geht natürlich darum, die Zahl mit möglichst wenig Versuchen zu erraten. Hier noch die Tabelle der Variablen:

S:	Anzahl der Stellen
E,E():	Eingabe, Ziffern der Eingabe
L():	Ziffern der Lösung
B(),C():	Belegungsvektoren für E bzw. L
R:	Richtige Ziffern an der richtigen Stelle
F:	Alle richtigen Ziffern
V:	Versuche
B,I,J:	Hilfs- und Laufvariablen

(Hans Haberl/aa)

```
1 INPUT"ᴝSTELLEN";S:E=INT(10↑S*RND(0)):GOSUB2:FORI=1TOS:L(I)=E(I):NEXT:GOTO3
2 FORI=1TOS:E(I)=E-10*INT(E/10):E=INT(E/10):C(I)=0:B(I)=0:R=R-(L(I)=E(I)):NEXT:R
ETURN
3 V=V+1:INPUTE:R=0:F=0:GOSUB2:FORI=1TOS:FORJ=1TOS:B=(L(J)=E(I))ANDNOTB(I)ANDNOTC
(J)
4 B(I)=B(I)+B:C(J)=C(J)+B:F=F-B:NEXTJ,I:PRINT"ᴝ"TAB(16)R"  "F-R"  "V:IFR<STHEN3
```

Programmierter Direktmodus

Programmierter Direktmodus hört sich wie ein Widerspruch in sich an. Entweder man befindet sich im Direktmodus oder es läuft ein Programm, beides gleichzeitig scheint kaum vereinbar. Dennoch gibt es eine Verbindung, die bisher ungeahnte Möglichkeiten eröffnet.

Das Geheimnis dieser Verbindung liegt im Tastaturpuffer (Tabelle 1) und dem Umstand, daß der Computer, nachdem er im Programm auf eine END-Anweisung trifft, so viele Zeichen aus dem Tastaturpuffer holt, wie der Anzahlspeicher angibt und sofort ausführt. Dies kann man sich zunutze machen, indem man mittels PRINT eine Anweisung auf den Bildschirm bringt, den Cursor veranlaßt, in diese Zeile zu springen und den Computer durch ein im Puffer abgelegtes RETURN mit der Abarbeitung der eingegebenen Bildschirmzeile fortfahren läßt. Dabei werden alle Zeichen über ihren ASCII-Code in den Puffer gebracht. Ein kleines Beispiel soll dieses Vorgehen verdeutlichen:

Geben Sie Beispiel 1 ein.

Was passiert? — Der Computer geht nach der END-Anweisung in den Direktmodus über und führt die beiden Steuerzeichen CURSOR/HOME und RETURN (CHR$(19 beziehungsweise 13)) aus. Der Cursor springt also in die erste Bildschirmzeile, wo zu lesen ist:

I=I+1 : GOTO 20

Diese Zeile führt der Computer nun aus und springt, nachdem er die Variable I um 1 hochgezählt hat, zurück in Programmzeile 20. Jetzt hilft nur noch RUN/STOP.

Beispiel 2

Dieses Programm bewirkt folgendes:

— Zeile 100 wird in das Programm eingefügt
— Zeile 350 wird aus dem Programm gelöscht
— der in Zeile 70 stehende Spruch wird durch einen — meiner Meinung nach treffenderen — ersetzt
— das geänderte Programm wird gelistet.

Anstelle des LIST-Befehles könnte auch ein GOTO beziehungsweise GOSUB-Befehl wieder in das Programm zurückspringen. Allerdings ist auch hier zu beachten, daß, sobald man neue Basic-Zeilen einfügt (beziehungsweise löscht), die Variablenwerte verloren gehen.

Mittels dieser Methode kann man zum Beispiel ein Programm entwickeln, das die DATA-Zeilen eines Sprites berechnet, ins laufende Programm übernimmt und dann die restlichen Programmzeilen herauslöscht, so daß nur ein Sprite-Ladeprogramm übrigbleibt, welches sofort abgespeichert werden kann. Eine weitere sinnvolle Anwendungsmöglichkeit des »Programmierten Direktmodus« können Sie den folgenden Ausführungen entnehmen.

System Lademenü

Das System soll die folgenden Aufgaben erfüllen:

(1) Mit verschiedenen Programmen (eventuell auf verschiedenen Disketten) arbeiten, ohne daß ständig LOAD und RUN gegeben werden muß.

(2) Den ärgerlichen »file not found error« verhindern, der bereits auftritt, wenn man ein Leerzeichen zuviel oder zuwenig eingibt.

(3) Programmiertes Aufrufen von Programmen (eventuell mit Parameterübergabe). Arbeiten mit dem Programmsystem ohne genaue Kenntnis der verwendeten Filenamen.

Die Idee

Mit Hilfe des Commodore-Programmes DOS 5.1 und der Methode des Programmierten Direktmodus kann man das oben genannte Ziel erreichen. Man geht dabei folgendermaßen vor:

Auf jede Diskette, die nach diesem System arbeiten soll, bringt man das Programm DOS 5.1 sowie dessen Lader unter einem möglichst kurzen einprägsamen Filenamen (hier »£«-Listing 1). Weiterhin kopiert man das Programm »Lademenü« auf jede der Disketten und trägt in dieses die Filenamen ein (Listing 2). Dabei ist darauf zu achten, daß die Filenamen »£« und »Lademenü« überall exakt gleich sind.

Der Arbeitsablauf gestaltet sich dann in folgender Weise: Nach dem Einschalten von Computer und Floppy legt man die gewünschte Diskette ein und lädt »£«. Dieses Programm initialisiert nach dem Starten die bekannten DOS 5.1-Befehle. Das

Beispiel 1

```
10 I=1
20 PRINT"(shift-clr/home)( 6xCursor down)"I". Lauf"
30 FOR J=0 TO 1500 : NEXT
40 PRINT"(shift-clr/home) I=I+1 : GOTO 20"
50 FOR J=0 TO 1500 : NEXT
60 POKE 631,19 : POKE 632,13 : POKE 198,2 : END
```

Beispiel 2

```
10 PRINT"(shift-clr/home)";
20 PRINT 100 "REM == Zeile 100 wird eingefügt =="
30 PRINT 350
40 PRINT"70 REM Morgenstund hat Blei im Hintern"
50 PRINT"LIST"
60 POKE 631,19 : FOR I=1 TO 5 : POKE 632 + I,13 :
NEXT
65 POKE 198,6 : END
70 REM Morgenstund hat Gold im Mund
350 REM == Diese Zeile wird gelöscht!==
```

Tabelle 1

	Speicherstelle (n) dezimal	hex
Tastaturpuffer	631 — 640	0277 — 0280
erweitert*	631 — 645	0277 — 0285
Anzahlspeicher	198	00C6
max. Größe des Tastaturpuffers	649	0289

* Anm.: Obwohl der Tastaturpuffer normalerweise nur 10 Zeichen faßt, können insgesamt 15 Zeichen dort abgelegt werden.

```
1 REM ^ = PFEIL NACH OBEN
2 REM _ = PFEIL NACH LINKS
10 POKE53281, 0:POKE53280, 0:PRINTCHR$(1
58):PRINTCHR$(142)
20 A=0:PRINT"♥":PRINT"                ────────
"
22 PRINT"               ▧LADEMENUE"
25 PRINT"              ────────────":PRIN
T:PRINT
30 PRINT"          1 = LISTE DER DOS BE
FEHLE
31 PRINT"          2 =BEISPIELPROGRAMM"
32 PRINT"          3 =*HIER MUESSEN
33 PRINT"          4 =*SIE IHRE
34 PRINT"          5 =*PROGRAMME EIN-
36 PRINT"          6 =*TRAGEN
37 PRINT"          7 =*
38 PRINT"          8 =*
39 PRINT"          9 =*
40 PRINT"         10 = ENDE":PRINT
100 INPUT"IHRE WAHL";A:A=INT(A):IFA<1ORA
>10THENPRINT"♥":GOTO20
105 PRINT"♥";
110 ONAGOTO300,111,112,113,114,115,116,1
17,118,119,
111 PRINT"^BEISPIELPRG"           :GOTO200
112 PRINT"^PRG2"                  :GOTO200
113 PRINT"^PRG3"                  :GOTO200
114 PRINT"^PRG4"                  :GOTO200
115 PRINT"^PRG4"                  :GOTO200
116 PRINT"^PRG5"                  :GOTO200
117 PRINT"^PRG6"                  :GOTO200
118 PRINT"^ UND SO WEITER       ":GOTO200
119 POKE198,0:PRINT"CIAO":END
120 POKE198,0:END
200 POKE631,19:POKE632,13:POKE198,2:END
300 PRINT:PRINT"DER DOS MANAGER BIETET F
OLGENDE BEFEHLE:":PRINT
310 PRINT"================================
=========":PRINT
320 PRINT"        = SAVE
330 PRINT"      / = LOAD
340 PRINT"      ^ = LOAD MIT AUTOSTART
350 PRINT"      @= ANZEIGEN DISKSTATUS
360 PRINT"     @$ = ANZEIGEN DIREKTORY
370 PRINT"================================
=========":PRINT
380 PRINT"DIESE BEFEHLE KOENNEN SIE JETZ
T NUTZEN!"
390 PRINT"================================
=========":POKE198,0
395 PRINT:PRINT:PRINT"               ▧*T
ASTE*":WAIT198,1:GETA$:GOTO20
READY.
```

Listing 2. »Lademenü«

```
10 REM ^ = PFEIL NACH OBEN
20 POKE53280,0:POKE53281,0:PRINTCHR$(158
)
30 IFA=0THENA=1:LOAD"DOS 5.1",8,1
40 IFA=1THENSYS12*4096+12*256
50 PRINT"♥";"^LADEMENUE"
60 POKE631,19:POKE632,13:POKE198,2:END
READY.
```

Listing 1. »£«

```
1 REM ^ = PFEIL NACH OBEN
100 REM BEISPIELPROGRAMM
110 :
120 BERNHARD LAUER
130 :
140 PRINT"♥"
150 PRINT"AN JEDES LISTING MUESSEN SIE"
160 PRINT"DIE ZEILE":PRINT
170 PRINT"PRINT CHR$(147);"CHR$(34)"^LAD
EMENUE"CHR$(34)":POKE 631,19:";
180 PRINT"POKE 632,13:POKE198,2:END":PRI
NT
190 PRINT"ANSTELLE DES END ANFUEGEN!"
200 PRINT:PRINT:PRINT:PRINT
210 PRINT"▧ ** TASTE **"
220 POKE 198,0:WAIT 198,1
230 PRINT"♥";"^LADEMENUE":POKE631,19:POK
E632,13:POKE198,2:END
READY.
```

Listing 3. »Beispielprogramm«

RUN für »£« soll auch gleichzeitig das letzte sein, da nun der DOS-Befehl »↑« zur Verfügung steht. Mit dessen Hilfe wird das Lademenü dieser Disk geladen und gestartet, welches die Programme anbietet und automatisch richtig lädt und startet. Zusätzlich bietet das Lademenü auch einen Überblick über die wichtigsten DOS-Befehle, die ja nun zur Verfügung stehen.

Um den Kreis zu schließen wird an jedes Programm auf der Diskette anstelle des üblichen END die folgende Zeile eingefügt:

PRINT CHR$(147); »Lademenü« :POKE 631,19:POKE 632,13:POKE 198,2:END

Dadurch wird nach jedem regulären Programmabbruch wieder das Lademenü dieser, oder wenn die Diskette vorher gewechselt wurde, einer beliebigen anderen Disk geladen. Nun wird durch das Lademenü das nächste ausgewählte Programm geladen und gestartet. Will man Parameter an andere Programme übergeben, so erreicht man dies über sequentielle Dateien.

Die Lösung

- ☐ Listing »£«
- ☐ Listing »Lademenü«
- ☐ Listing »Beispielprogramm«

Ausblick

Schreibt man noch einen Autostart für »£« (eventuell mittels des Beispiels in Ausgabe 6/84) so kann man das »System Lademenü« zu einem System »Nie wieder RUN« ausbauen.

(Bernhard Lauer/rg)

Automatische Zeilennumerierung

Das lästige Durchnumerieren der Zeilen bei der Programmierung kann Ihnen dieses kleine Programm abnehmen.

Die Syntax des AUTO-Befehls ist:
← A anfangszeilennummer, schrittweite
Nach Eingabe dieses Befehls wird die Zeilennummer vorgegeben und nach RETURN um »schrittweite« erhöht.

Um aus dem AUTO-Modus wieder herauszukommen, muß man nach Vorgabe einer Zeilennummer
»←« RETURN eingeben.

Falls man nach Vorgabe einer Zeilennummer die RETURN-Taste betätigt, wird die entsprechende Zeile, falls sie vorhanden ist, gelöscht. Hiermit lassen sich auch sehr schnell Programmblöcke löschen, falls man die RETURN-Taste gedrückt hält, die Zeilenvorgabe weiterläuft und die entsprechenden Zeilennummern gelöscht werden.

»←«=CHR$(95)
»A«=CHR$(65)

Das Programm als Basic-Lader eintippen, anschließend mit RUN starten. Falls »FEHLER IN DEN DATAZEILEN« erscheint, DATAs auf Tippfehler überprüfen. Falls »OK«, kann die Basic-Erweiterung mit SYS 49152 initialisiert werden. Nun hat man das Basic um den Befehl »A« erweitert.

(Frank Siedel/rg)

```
1010 rem*******************************
1020 rem**        auto fuer c 64     **
1030 rem**            von            **
1040 rem**        frank siedel       **
1050 rem**        posener str. 18    **
1060 rem**        2945 sande         **
1070 rem*******************************
1080 :
1090 :
1100 :
1110 :data169,11,141,8,3,169,192,141,9,3
,96,32,115,0,8,201,95,240,4,40,76,231
1120 :data167,32,115,0,201,65,208,245,32
,115,0,24,32,107,169,165,20,133,38
1130 :data165,21,133,39,32,253,174,24,32
,107,169,165,20,133,40,165,21,133,41
1140 :data169,129,141,2,3,169,192,141,3,
3,169,128,141,138,2,165,39,133,98,165
1150 :data38,133,99,162,144,56,32,73,188
,32,221,189,162,0,189,1,1,240,9,157
1160 :data0,2,32,210,255,232,208,242,32,
18,225,201,95,240,30,201,13,240,45
1170 :data157,0,2,232,32,98,165,76,134,1
64,24,165,38,101,40,133,38,165,39,101
1180 :data41,133,39,76,75,192,169,131,14
1,2,3,169,164,141,3,3,169,0,141,138
1190 :data2,40,76,116,164,32,118,165,76,
134,164
1200 :
1210 :
1220 printchr$(147)
1230 su=0
1240 fori=1to170
1250 reada
1260 su=su+a
1270 poke49151+i,a
1280 next
1290 ifsu<>17417 then print "fehler in d
en datazeilen":end
1300 print"ok":end

ready.
```

Musik aus der Datasette

Mit dieser Routine ist es beim C 64 möglich, die Datasette ohne technichen Umbau als normalen Kassettenrecorder zu betreiben.

Die Maschinenroutine, die mit SYS 49152 gestartet wird, stellt eine Endlosschleife dar, die mit der SHIFT-Taste jederzeit abgebrochen werden kann. In der Schleife selber werden die Datenbits aus der Datasette kommend von dem Interrupt-Control-Register der CIA 1 isoliert. Der Lautsprecher im Fernseher oder Monitor wird dem logischen Zustand des einzelnen Datenbits entsprechend ein- oder ausgeschaltet. Die für uns daraus resultierende Frequenz wird originalgetreu wiedergegeben, nur nicht die Tonqualität. Der Vorteil der Maschinenroutine ist, daß von der Datasette nichts geladen, sondern nur wiedergegeben wird. Es kann also bei einer Kassette irgendwo »hineingehört« werden. Als Alternative zu Programmen sollte man zur einfachen Musikkassette greifen. Mit Phantasie kann die Musik erkannt werden, doch die Tonqualität läßt noch zu wünschen übrig.

Programm

```
10   FOR I=49152 TO 49178:READA:POKEI,A:NEXT
20   SYS 49152
30   DATA 173,142,2,240,1,96,173,13,220,41,16,240,7,169,15,
     141,24,212,144,236
40   DATA 169,0,141,24,212,144,229
```

(Jörg Wagner/rg)

List- und Löschschutz leicht gemacht

Es wurden schon viele Methoden veröffentlicht, um ein Basic-Programm gegen Listen zu schützen. Aber alle mir bekannten Möglichkeiten weisen entschiedene Nachteile auf. Entweder der Schutz ist nicht sicher genug und leicht zu entfernen, oder er ist viel zu aufwendig.

Ich habe mich daher entschlossen, ein Programm zu schreiben, das diese Mängel umgeht und sogar noch andere positive Merkmale aufweist.

Zunächst eine Zusammenfassung von drei mir bekannten Listschutzmöglichkeiten mit ihren Vor- und Nachteilen:

1. Möglichkeit

In die erste Zeile des Basic-Programms (zum Beispiel Zeilennummer 1) wird REM, gefolgt von zwei Anführungszeichen und SHIFT L, geschrieben.

1 REM""L (RETURN)

Der Cursor wird nun auf das zweite Anführungszeichen gesetzt und sechsmal SHIFT INST gedrückt (das Anführungszeichen wird um sechs Positionen nach rechts geschoben). Dann wird sechsmal DEL eingegeben (es erscheinen als Steuerzeichen sechs reverse T) und die Zeile mit (RETURN) abgespeichert. Wird nun der LIST-Befehl aufgerufen, meldet sich der Rechner mit:

"

?SYNTAX ERROR

READY.

Auf den ersten Blick sehr beeindruckend, aber durch Entfernen dieser Zeile ist der Listschutz wieder aufgehoben. Außerdem ist ein 'LIST 2' noch möglich.

2. Möglichkeit

In jede Basic-Zeile werden synthetische Steuerzeichen eingefügt (genaue Beschreibung im 64'er-Magazin, Ausgabe 6/84). Diese Methode ist zwar recht sicher, will man aber alle Zeilen eines längeren Basic-Programms schützen, ist der Aufwand viel zu groß, vom Speicherplatzbedarf der Steuerzeichen ganz abgesehen.

3. Möglichkeit

Durch POKE 775,200 wird der Listbefehl außer Kraft gesetzt, durch POKE 775,167 wird diese Wirkung wieder aufgehoben. Dieser Listschutz ist zwar wirkungsvoll, aber er muß erst durch diesen POKE-Befehl aktiviert werden. Nach dem Laden eines Programms ist er daher noch nicht aktiv.

Das hier vorgestellte Programm erzeugt nicht nur einen sicheren Listschutz, sondern schützt auch vor dem Löschen einzelner Basic-Zeilen. So können zum Beispiel Hinweise auf ein Kopierrecht und auf den Autor eines Programms nicht geändert oder entfernt werden. Auch kann ein so gesichertes Programm nur mit RUN gestartet werden, ein RUN, gefolgt von einer Zeilennummer, führt zu einer Fehlermeldung. Jede Zeile des Programms ist geschützt, es können also auch einzelne Zeilen nicht gelistet werden. Einzige Bedingung für die Verwendung dieses Schutzes: Das zu schützende Programm darf keine Zeilennummern 0 und 1 enthalten. Ansonsten wird eine Fehlermeldung ausgegeben und das Programm bleibt unverändert.

Das Listschutzprogramm liegt als Basic-Lader vor. Nachdem es richtig abgetippt wurde, kann es durch RUN gestartet werden. Das Maschinenprogramm steht dann im Speicher ab der Adresse 50000 zur Verfügung. Das zu schützende Basic-Programm kann nun geladen werden, durch SYS 50000 wird das Schutzprogramm aktiviert und das Basic-Programm geschützt. Es kann nun wieder auf Kassette/Diskette gespeichert werden. Das mit dem Listschutz versehene Programm ist nur um wenige Bytes größer als vorher.

Funktionsweise

Das Maschinenprogramm generiert zwei Basic-Zeilen mit den Zeilennummern 0 und 1. Die Zeile 0 ist eine REM-Zeile, in der ein unlistbares Zeichen (SHIFT L) steht. Hinter diesem Zeichen stehen dann noch zwei kurze Maschinenprogramme, deren Funktionen im folgenden noch erklärt werden. In der zweiten Zeile steht ein SYS-Befehl, der eine der beiden Maschinenroutinen in Zeile 0 startet. Sind diese beiden Zeilen nun erzeugt, wird die Zeilennummer 0 durch eine höhere, eigentlich unerlaubte Zeilennummer (größer 64000) ersetzt. Diese Zeile kann daher auch nicht gelöscht werden.

Da alle nun folgenden Zeilen des Programms kleiner sind als die erste, können diese vom Computer nicht mehr erkannt werden. Ein Sprung in eine solche Zeile führt zu der Fehlermeldung: ?UNDEF'D STATEMENT ERROR. Es kann daher auch keine Zeile gelöscht werden, da diese für den Computer ja nicht mehr vorhanden sind.

Der einzige Nachteil ist, daß es nicht nur ein perfekter List- und Löschschutz, sondern auch ein RUN-Schutz ist (auch Sprungziele innerhalb des Programms können nicht gefunden werden).

Wird das geschützte Programm gestartet, trifft der Interpreter als erstes auf den SYS-Befehl in Zeile 1. Es folgt ein Sprung in das Maschinenprogramm in der REM-Zeile. Dort wird die Zeilennummer wieder auf 0 gesetzt, und der Vektor auf den Basic-Warmstart wird auf die zweite Maschinenroutine gesetzt.

Nun kann das Basic-Programm ohne Fehler ausgeführt werden. Wird der Programmlauf unterbrochen (durch STOP-Taste, Fehlermeldungen, Programmende und so weiter), wird das zweite Maschinenprogramm über den Basic-Warmstartvektor angesprungen. Dort wird die Zeilennummer wieder hochgesetzt, der Warmstartvektor wieder auf den normalen Wert gebracht und die Warmstartroutine angesprungen. Das Programm liegt nun wieder in der geschützten Form vor.

(Ulrich von Gaisberg/rg)

```
0 rem **********************************
1 rem *         u. v. gaisberg         *
2 rem *         am zuckerberg 70       *
3 rem *         7140 ludwigsburg       *
4 rem *         tel. 07141/55910       *
5 rem **********************************
6 fori=0to340:reada:b=b+a:poke50000+i,a
7 next i
8 if b <> 33527 then print"fehler in dat
as !":end
9 print"ok !":end
10 rem datas fuer maschinenprogramm
11 data169,0,141,32,208,141,33,208,169,1
,141,134,2,32,68,229,174,3,8,172
12 data4,8,192,0,208,7,224,2,176,3,76,20
6,195,162,0,142,134,2,169,32,32
13 data210,255,232,224,50,208,246,162,0,
189,21,196,157,0,4,232,224,29,208
14 data245,169,24,157,0,4,232,224,69,208
,246,162,0,189,50,196,157,80,4,232
15 data224,8,208,245,162,0,189,58,196,15
7,120,4,232,224,8,208,245,162,10
16 data160,0,24,32,240,255,169,19,162,13
,160,4,141,119,2,142,120,2,142,121
17 data2,142,122,2,132,198,96,162,0,189,
99,196,32,210,255,232,224,31,208
18 data245,96,32,68,229,162,10,160,0,24,
32,240,255,162,1,142,134,2,202,189
19 data66,196,32,210,255,232,224,33,208,
245,169,20,162,17,160,255,141,18
20 data8,142,29,8,140,4,8,162,0,189,130,
196,157,32,8,232,224,34,208,245
21 data96,48,18,5,13,34,148,148,148,148,
148,148,148,148,148,148,34,12,12
22 data9,19,20,19,3,8,21,20,26,26,76,49,
19,25,19,50,48,57,56,19,25,19,53
23 data48,49,52,48,80,82,79,71,82,65,77,
77,32,45,32,40,67,41,32,85,46,86
24 data46,71,65,73,83,66,69,82,71,32,32,
49,57,56,52,66,73,84,84,69,32,90
25 data69,73,76,69,32,48,32,85,78,68,32,
49,32,69,78,84,70,69,82,78,69,78
26 data32,33,169,255,141,4,8,169,131,162
,164,141,2,3,142,3,3,76,131,164
27 data165,2,141,4,8,169,32,162,8,141,2,
3,142,3,3,96,0

ready.
```

Programm für List- und Lösch-Schutz

Stringy: C64-Erweiterung

Stringy stellt eine Basic-Interpretererweiterung dar, die den Befehlssatz des C 64 um acht Befehle ergänzt. Mit diesen Befehlen ausgestattet, kann man sich einen Programmgenerator von Basic aus programmieren.

Das Listing zu Stringy entstand mit Hilfe von Stringy. Dabei wurden die Zahlen formatiert, die Prüfsummen berechnet und nach jeder vierten Zeile angefügt. Mit Stringy kann man Strubs-ähnliche Erweiterungen programmieren (Der Grund, weshalb ich Stringy schrieb). Man könnte auch ein Programm schreiben, das die in einem Basic-Programm vorkommenden Grafikzeichen durch die entsprechenden CHR$-Funktionen ersetzt, damit sie im Listing besser zu erkennen sind. Auch Sprite- oder Bildschirmmasken-Generatoren sind recht einfach zu programmieren. Der wichtigste Befehl von Stringy ist der !INPUT-Befehl. Mit ihm kann man einen String, der eine Basic-Zeile mit Zeilennummer darstellt, bei laufendem Programm in das Basic-Programm übernehmen — ohne, daß dabei die Programmausführung unterbrochen wird.

Umgekehrt kann es sinnvoll sein, eine Zeile aus dem Basic-Programm herauszuholen, um sie einer Stringvariablen zuzuordnen. Dies ermöglicht der !GET-Befehl.

Damit es keine Komplikationen mit den Basic-Zeilennummern gibt, teilt der !NEXL-Befehl Ihnen die Folge der Zeilennummern mit.

Die anderen fünf Befehle dienen der Stringverarbeitung. Vier davon sind dem Sinn nach identisch mit den entsprechenden Stringoperationen aus Simons Basic, mit dem Unterschied, daß die Parameter beliebig komplizierte Ausdrücke sein können (dies gilt für alle Befehle von Stringy).

Der letzte der fünf Stringbefehle ist der !REPLACE-Befehl.

Die Stringy-Befehle

Nachfolgend bedeuten str1, str2, str3 immer Stringausdrücke und m, n, p, w, z immer numerische Ausdrücke.

!PLACE

Format: !PLACE (str1,str2)
!PLACE (str1,str2,m)
!PLACE (str1,str2,m,n)

Funktion: Bestimmung der Position, an der str2 in str1 steht. Die Angabe von m und n grenzt str1 auf einen Teilstring ein. Nur dieser Teilstring von str1 wird dann durchsucht, und nicht der ganze String. m gibt den Beginn dieses Teilstrings an, gerechnet vom Anfang von str1, n bestimmt das Ende des Teilstrings. Vorsichtig: n wird vom Ende von str1 aus gezählt, also in anderer Richtung als m.
Beispiel: »PRINT !PLACE ("COMMODORE", "O")« liefert 2 als Antwort.
»PRINT !PLACE ("COMMODORE", "O",3)« liefert 5 als Antwort, da nur in "MMODORE" gesucht wurde
»PRINT !PLACE ("COMMODORE", "E", 1, 4)« liefert 0 als Ergebnis, da "E" nicht in dem Teilstring "COMMOD" enthalten ist.

!REPLACE

Format: !REPLACE (str1,str2,str3)
!REPLACE (str1,str2,str3,m)
!REPLACE (str1,str2,str3,m,n)

Funktion: Ersetzen aller str2, die in str1 vorkommen, durch str3. Dabei kann str1, wie beim !PLACE-Befehl beschrieben, durch n und m eingegrenzt werden.
Beispiel: 10 A$="INDEX=B$+C$"
20 B$=!REPLACE(A$,"INDEX","IN$")
Nach Ausführung gilt: B$="IN$=B$+C$"
10 N$="PETER PAUL MARY"
20 M$=!REPLACE (N$,"PETER", " ")
Nach Ausführung gilt: M$="PAUL MARY"

!INSERT

Format: !INSERT (str1,str2,p)
Funktion: Fügt str2 in str1 ein. Die Position p bestimmt, an welcher Stelle str2 in str1 eingefügt werden soll.
Ist dabei p=0 oder p=len (str1), so wird angefügt.
Beispiel: »PRINT !INSERT ("ABCEF","D",3)« liefert: "ABCDEF"
»PRINT !INSERT ("ABCEF","D",0)« liefert: "DABCEF"
»PRINT !INSERT ("ABCEF","D",5)« liefert: "ABCEFD"

!STOVER

Format: !STOVER (str1,str2,p)
Funktion: Überschreibt str1 mit str2.
Die Position, ab der str1 überschrieben werden soll, wird durch p angegeben.
Ist str2 länger als str1, oder ist wegen der Positionsangabe p ein Überschreiben nicht möglich, so erfolgt ein ILLEGAL QUANTITY ERROR.
Beispiel: »PRINT !STOVER ("GOTO XXXX", "0169",6)« liefert: "GOTO 0169"

!DUP

Format: !DUP (str,w)
Funktion: Es wird str w-mal dupliziert.
Beispiel: A$=!DUP(".",255) liefert einen String mit 255 einzelnen Punkten.

!INPUT

Format: !INPUT(str)
Funktion: Hat str keine Zeilennummer am Anfang, so geht der Computer in den Direktmodus über und führt str sofort aus. Soll der Computer anschließend zum Programm zurückkehren, so muß der letzte Befehl in str ein »GOTO (Zeilennummer)« sein.

Beginnt str mit einer Zeilennummer, so wird str als Basic-Zeile in das laufende Programm eingefügt, sofern in dem Programm nicht bereits eine Zeile mit derselben Zeilennummer existiert. Andernfalls wird die betreffende Zeile vor dem Einfügen gelöscht. Wenn allerdings diese zu löschende Zeile eine noch offene FOR...TO-Anweisung oder ein noch nicht durch RETURN abgeschlossenes GOSUB enthält, so erfolgt ein CAN'T CONTINUE ERROR.

Die gleiche Fehlermeldung erscheint auch, wenn Sie eine Zeile löschen wollen, in der sich der DATA-Zeiger momentan befindet. Beispiel:
10 DATA56
20 READA:!INPUT(STR$(10))
Nach RUN erfolgt ein CAN'T CONTINUE ERROR, da sich der DATA-Zeiger in Zeile 10 befindet. Durch einen RESTORE-Befehl, läßt sich diese Zeile dennoch löschen:
10 DATA56
20 READA:RESTORE:!INPUT("10")
Nach Ausführung dieser beiden Zeilen ist die Zeile 10 gelöscht. Enthielt die gelöschte Basic-Zeile eine DEF-Anweisung, so gilt diese Funktion als nicht definiert. Enthielt die gelöschte Basic-Zeile eine Stringvariablenzuordnung der Art »AA$="ABCD"« oder »A$(n)="ABCDE"«, so ist anschließend die Variable nur noch als Leerstring definiert.
Soll der !INPUT-Befehl direkt nach einem THEN stehen, dann ist ein Doppelpunkt einzufügen »...THEN:!INPUT...«

!GET

Format: !GET(z)
Funktion: Es wird die Basic-Zeile mit der Zeilennummer z in Stringformat geholt. Der Parameter z darf dabei nicht den Wert 0 haben. Beispiel:

```
10  REM !GET-DEMO
20  PRINT!GET(10):PRINT!GET(20)
30  A$=!GET(30)
40  PRINT MID$(A$,!PLACE(A$," ")+1)
```

!NEXL

Format: !NEXL(z)
Funktion: Es wird die auf z folgende Basic-Zeilennummer geholt. Hat !NEXL(z) den Wert 0, so bedeutet dies, daß auf z keine Basic-Zeilen mehr folgen. Beispiel:

```
10  REM !NEXL-DEMO
20  REM SIMULATION DES LIST-BEFEHLS
30  Z=0
40  Z=!NEXL(Z):IF Z=0 THEN END
50  PRINT !GET(Z):GOTO 40
```

Zum Schluß noch einige Daten zu Stringy. Stringy belegt den Speicher von $c100 bis $c85a. Der Bereich von $c000 bis $c0ff dient als Stringzwischenspeicher (je nach Befehl wird dieser Raum benutzt). Die Speicherplätze $c85b bis $c865 dienen als Zwischenspeicher für einige wichtige Betriebssystemdaten. Der unter dem Basic-ROM liegende Speicherbereich wird mitbenutzt.

(Karl Szameitat/ev)

```
0 REM STRINGY BY KARL SZAMEITAT,MUEHLENSTR 8B,
   2355 WANKENDORF
100 DATA 238,000,192,208,003,076,113,165,238,
   017,193,208,003,238,018,193
101 DATA 141,071,192,096,165,001,041,254,133,
   001,096,165,001,009,001,133
102 DATA 001,096,169,000,162,160,133,020,134,
   021,133,002,162,192,160,000
103 DATA 177,020,145,020,200,208,249,230,021,
   228,021,208,241,169,096,141,7522
104 DATA 020,167,141,038,181,160,005,169,234,
   153,209,166,136,016,250,160
105 DATA 002,153,237,164,136,016,250,169,000,
   162,193,141,072,171,142,073
106 DATA 171,169,243,141,000,003,142,001,003,
   169,122,141,008,003,142,009
107 DATA 003,169,170,141,010,003,142,011,003,
   096,032,115,000,008,201,033,6890
108 DATA 240,004,040,076,231,167,032,115,000,
   201,133,208,004,104,076,015
109 DATA 196,234,234,234,234,234,234,234,234,
   234,234,234,234,234,234,234
110 DATA 234,234,234,234,234,234,234,076,008,
   175,169,000,133,013,032,115
111 DATA 000,008,201,033,240,004,040,076,141,
   174,104,032,115,000,160,006,9045
112 DATA 217,077,200,240,005,136,016,248,048,
   199,185,084,200,168,032,115
113 DATA 000,217,032,200,208,187,200,201,040,
   208,243,185,032,200,072,185
114 DATA 033,200,072,076,115,000,169,000,162,
   192,141,017,193,142,018,193
115 DATA 076,016,193,138,048,010,169,000,133,
   002,032,027,193,076,058,164,7638
116 DATA 076,116,164,032,020,193,134,035,104,
   133,020,104,133,021,169,006
117 DATA 032,251,163,230,002,160,011,185,165,
   000,072,136,016,249,165,021
118 DATA 072,165,020,072,166,035,134,174,169,
   165,162,000,133,175,134,176
119 DATA 032,158,173,032,143,173,160,002,177,
   100,145,175,136,016,249,165,7306
120 DATA 100,164,101,032,219,182,198,174,240,
   011,032,253,174,230,175,230
121 DATA 175,230,175,208,219,162,001,032,121,
   000,201,041,240,006,032,253
122 DATA 174,032,158,183,134,174,162,001,201,
   041,240,006,032,253,174,032
123 DATA 158,183,134,175,076,247,174,032,230,
   193,165,073,072,165,074,072,8831
124 DATA 032,138,173,032,247,183,032,247,174,
   032,019,166,032,020,193,032
125 DATA 189,166,032,027,193,104,133,074,104,
   133,073,169,001,162,192,133
126 DATA 111,134,112,173,017,193,076,192,180,
   166,168,165,166,197,175,240
127 DATA 023,160,000,177,166,209,169,208,006,
   202,240,011,200,208,244,230,8555
128 DATA 166,208,230,230,167,208,226,024,096,
   165,174,240,025,165,175,240
129 DATA 021,198,174,198,175,024,165,174,101,
   175,176,010,101,168,176,006
130 DATA 197,165,240,005,144,003,076,072,178,
   024,165,166,101,165,056,229
131 DATA 175,056,229,168,133,175,230,175,024,
   165,166,101,174,133,166,144,9181
132 DATA 002,230,167,096,162,002,032,003,194,
   165,168,240,034,165,165,240
133 DATA 030,197,168,144,026,165,166,072,032,
   201,194,032,169,194,104,176
134 DATA 014,229,166,073,255,170,232,032,051,
   195,138,168,076,162,179,162
135 DATA 000,240,244,104,133,020,104,133,021,
   160,244,104,153,177,255,200,8729
136 DATA 208,249,165,021,072,165,020,072,198,
   002,208,003,032,027,193,096
137 DATA 162,003,032,003,194,024,165,166,133,
   034,101,165,133,037,165,167
138 DATA 133,035,032,230,193,032,201,194,032,
   169,194,144,004,165,037,133
139 DATA 166,165,166,197,034,240,015,160,000,
   177,034,032,000,193,230,034,7186
140 DATA 208,239,230,035,208,235,165,034,197,
   037,240,037,024,160,000,165
141 DATA 034,101,168,133,034,144,002,230,035,
   196,171,240,008,177,172,032
142 DATA 000,193,200,208,244,165,034,197,037,
   240,006,032,191,194,076,107
143 DATA 195,032,051,195,076,155,194,162,002,
   032,003,194,165,168,240,008,8087
144 DATA 165,165,240,004,197,174,176,003,076,
   072,178,024,101,168,176,248
145 DATA 032,125,180,138,208,001,136,202,142,
   017,193,140,018,193,160,000
146 DATA 140,000,192,196,174,240,014,177,166,
   032,000,193,200,198,165,208
147 DATA 242,165,168,240,020,152,170,160,000,
   177,169,032,000,193,200,198,8633
148 DATA 168,208,246,138,168,165,165,208,222,
   032,051,195,076,202,180,160
149 DATA 011,185,165,000,153,091,200,136,016,
   247,032,166,179,032,115,000
150 DATA 032,250,174,032,158,173,032,247,174,
   032,163,182,201,089,144,003
151 DATA 076,113,165,170,208,003,076,055,198,
   165,122,164,123,133,165,132,8466
152 DATA 166,160,000,132,167,132,168,132,169,
   232,202,240,008,177,034,153
153 DATA 000,002,200,208,245,138,153,000,002,
   202,160,001,134,122,132,123
154 DATA 032,115,000,144,003,076,019,200,032,
   107,169,032,121,165,132,011
155 DATA 032,019,166,176,003,076,216,196,032,
   072,198,176,036,166,020,164,7200
156 DATA 021,196,064,208,020,228,063,208,016,
   165,043,229,065,008,201,001
157 DATA 208,015,040,165,044,229,066,208,008,
   196,058,208,009,228,057,208
158 DATA 005,162,026,076,055,164,165,095,072,
   165,096,072,056,160,000,177
159 DATA 095,133,167,229,095,133,169,200,177,
   095,133,168,032,131,198,162,7546
160 DATA 000,056,181,045,229,169,149,045,232,
   176,002,214,045,232,224,006
161 DATA 208,239,104,133,096,104,133,095,173,
   000,002,208,005,133,011,076
```

Listing »Stringy«

```
162 DATA 031,197,024,165,049,133,090,101,011,
    133,088,164,050,132,091,144
163 DATA 001,200,132,089,032,184,163,162,000,
    024,181,045,101,011,149,045,6847
164 DATA 232,144,002,246,045,232,224,004,208,
    239,165,020,164,021,141,254
165 DATA 001,140,255,001,164,011,136,185,252,
    001,145,095,136,016,248,032
166 DATA 051,165,165,165,164,166,032,182,198,
    133,122,132,123,165,065,164
167 DATA 066,032,182,198,133,065,132,066,032,
    206,198,165,045,166,046,133,8216
168 DATA 034,134,035,160,000,177,034,016,007,
    200,177,034,048,007,016,031
169 DATA 200,177,034,048,077,024,165,034,105,
    007,133,034,144,002,230,035
170 DATA 165,035,197,048,144,221,165,034,197,
    047,144,215,076,187,197,200
171 DATA 200,177,034,170,136,177,034,032,028,
    199,176,028,032,050,199,145,6648
172 DATA 034,200,138,145,034,200,200,177,034,
    170,136,177,034,032,050,199
173 DATA 145,034,138,200,145,034,208,189,169,
    000,168,145,034,200,145,034
174 DATA 208,179,200,200,200,177,034,170,136,
    177,034,032,028,199,176,232
175 DATA 032,050,199,145,034,200,138,145,034,
    208,154,165,047,166,048,133,8228
176 DATA 034,134,035,228,050,208,013,197,049,
    208,009,076,055,198,165,036
177 DATA 166,037,208,235,024,160,002,113,034,
    133,036,200,138,113,034,133
178 DATA 037,160,000,177,034,048,231,200,177,
    034,016,226,160,004,177,034
179 DATA 010,105,005,101,034,133,034,144,002,
    230,035,160,000,177,034,240,6620
180 DATA 022,200,200,177,034,170,136,177,034,
    032,028,199,176,034,032,050
181 DATA 199,145,034,200,138,145,034,024,165,
    034,105,003,133,034,144,002
182 DATA 230,035,165,035,197,037,144,211,165,
    034,197,036,144,205,176,158
183 DATA 169,000,168,145,034,240,224,160,244,
    185,103,199,153,177,255,200,8065
184 DATA 208,247,032,121,000,076,231,167,186,
    189,003,001,201,141,240,006
185 DATA 201,129,240,027,024,096,189,004,001,
    197,020,208,007,189,005,001
186 DATA 197,021,240,241,024,138,105,007,170,
    201,248,144,220,176,229,189
187 DATA 017,001,197,020,208,007,189,018,001,
    197,021,240,216,024,138,105,7736
188 DATA 018,208,229,160,000,177,095,133,034,
    200,177,095,133,035,165,049
189 DATA 133,036,165,050,133,037,136,177,034,
    145,095,165,036,197,034,208
190 DATA 007,165,037,197,035,208,001,096,230,
    034,208,002,230,035,230,095
191 DATA 208,229,230,096,208,225,196,096,144,
    013,240,012,229,169,176,001,7971
192 DATA 136,024,101,011,144,001,200,096,197,
    095,144,251,176,236,186,189
193 DATA 003,001,201,141,240,005,201,129,240,
    028,096,189,006,001,188,007
194 DATA 001,032,182,198,157,006,001,152,157,
    007,001,024,138,105,007,170
195 DATA 201,248,144,219,176,228,188,019,001,
    189,020,001,032,182,198,157,7404
196 DATA 020,001,152,157,019,001,189,004,001,
    188,005,001,032,182,198,157
197 DATA 004,001,152,157,005,001,024,138,105,
    018,208,211,228,168,144,006
198 DATA 208,014,197,167,176,010,228,096,144,
    007,208,005,197,095,176,001
199 DATA 024,096,228,052,240,004,176,023,144,
    004,197,051,176,017,228,096,6562
200 DATA 144,013,240,012,229,169,176,001,202,
    024,101,011,144,001,232,096
201 DATA 197,095,144,251,176,238,162,002,032,
    003,194,165,168,240,004,165
202 DATA 165,208,003,076,072,178,024,166,174,
    240,248,202,134,174,138,101
203 DATA 168,197,165,240,002,176,236,165,165,
    032,125,180,134,033,132,034,8518
204 DATA 164,165,136,177,166,145,098,152,208,
    248,024,165,174,101,033,133
205 DATA 033,144,002,230,034,164,168,136,177,
    169,145,033,152,208,248,032
206 DATA 051,195,076,202,180,076,227,168,162,
    001,032,003,194,165,165,208
207 DATA 003,076,072,178,032,230,193,166,174,
    240,246,160,000,177,166,032,8614
208 DATA 000,193,200,196,165,208,246,202,208,
    241,032,051,195,076,155,194
209 DATA 032,138,173,032,247,183,032,019,166,
    160,000,144,039,177,095,133
210 DATA 020,200,177,095,133,021,177,020,240,
    021,200,200,177,020,170,136
211 DATA 177,020,133,099,134,098,162,144,056,
    032,073,188,076,247,174,169,8321
212 DATA 000,170,240,238,200,177,095,240,246,
    166,095,165,096,134,020,133
213 DATA 021,208,215,169,255,133,058,032,121,
    165,032,115,000,076,055,198
214 DATA 040,194,118,076,065,067,069,040,195,
    003,069,080,076,065,067,069
215 DATA 040,195,079,078,083,069,082,084,040,
    195,182,164,086,069,082,040,7129
216 DATA 199,085,085,080,040,199,167,069,088,
    076,040,199,207,161,080,082
219 DATA 073,083,068,078,000,003,010,019,027,
    034,039,2291
300 :
310 REM POKE/PRUEFROUTINE
320 :
330 RESTORE:AD=49408:ZE=100
340 PR=0
350 READ PO:IF PO>255 THEN 370
360 PR=PR+PO:POKE AD,PO:AD=AD+1:GOTO 350
370 IF PO<>PR THEN PRINT"FEHLER«SPACE»IN«SPACE»
    ZEILEN"ZE"BIS"ZE+3:END
380 IF PO=2291 THEN PRINT"DATAS«SPACE»OK«SPACE»
    -«SPACE»SAVE«SPACE»PROGRAMM«SPACE»UND«SPACE»
    STARTE«SPACE»MIT«SPACE»SYS49442":END
390 ZE=ZE+4:GOTO 340
```

Listing »Stringy« (Schluß). In Zeile 370/380 bei »SPACE« einfach die Leertaste drücken!



Basic-Programmier-Handbuch

Wer sich endlich seine mehr oder weniger teure Computer-Anlage angeschafft hat, wird auch gerne die Kunst des Programmierens erlernen wollen. Die meisten Computerneulinge betätigen sich hier als Autodidakten, doch steht ihnen zumeist nur eine unzureichende Dokumentation im Handbuch zur Verfügung. Das nun in zweiter Auflage in deutscher Übersetzung erschienene »Basic-Programmier-Handbuch« aus der Reihe der Computer-Persönlich-Bücher ist ein weiteres Buch in der Masse der Literatur, die dem Neuling die Auswahl erschwert. Doch dieses Buch unterscheidet sich in einigen Punkten deutlich vom sonstigen Angebot.

Zunächst fällt einmal der Umfang dieser Einführung auf: Ganze 508 Seiten, in gut lesbarem Druck gesetzt, stehen dem Basic-Anfänger zur Durcharbeitung bevor. Doch das Arbeiten mit diesem Handbuch ist ausgesprochen leicht. Der Inhalt ist klar in Kapitel und Abschnitte unterteilt. Es beginnt mit einer allgemeinen Einführung in die Praxis der Programmierung, und über die Grundlagen der Programmierung bis hin zu den Höhen der Maschinencode-Programmierung wird dem Interessierten alles Wissenswerte vermittelt.

Im ersten Kapitel werden die Grundlagen einer Programmiersprache vermittelt, so die Ein- und Ausgabe und das Erstellen einfacher Listings. Im zweiten Kapitel werden dann Programmerstellung und -steuerung vorgestellt. Alles wird mit einfachen, durchweg »netten« Beispielen verdeutlicht. Überhaupt findet sich in diesem Buch nicht der Ernst, der das Arbeiten mit anderen Veröffentlichungen manchmal schnell verleidet. Einige humorstische Abbildungen und ein leichter Schreibstil lassen dieses Handbuch zu einer Lektüre werden, bei der jeder neue Abschnitt mit anhaltendem Interesse angegangen wird.

Am Ende jedes Kapitels stehen einige Fragen, die den Lernerfolg bestätigen sollen. Die in den ersten beiden Kapiteln vermittelten Grundkenntnisse werden im dritten Kapitel anhand eines Spielprogramms vertieft und mit einigen Raffinessen angereichert.

In die Feinheiten von Basic führen dann die Kapitel vier und fünf ein, Stringfunktionen und Programmierhilfen lernt der Leser ebenso kennen wie den Umgang mit der Peripherie. Nachdem schließlich das Kapitel sechs ebenso wichtige wie interessante Funktionen und Routinen für eine gute Programmierung, wie Fehler-, String- und Variablenbehandlung vorstellt, wird im letzten Kapitel ein komplettes Programm entwickelt. Hier kann der inzwischen zum »Fast-Profi« gewordene Leser alle Kenntnisse anwenden, um den Zauberwürfel auf seinem Computer zu simulieren.

Ein umfangreicher Anhang, in dem die verschiedenen Zahlensysteme sowie Tips zum Speichersparen und Programmbeschleunigen ebenso aufgelistet sind wie die Lösungen zu den Aufgaben am Ende jedes Kapitels schließt das Buch dann ab.

In diesem Basic-Programmierhandbuch wird kein spezieller Basic-Dialekt zugrunde gelegt, jedoch finden sich viele Befehle und Funktionen, die auf einem Kleinstcomputer wohl kaum anzutreffen sind. Der Umfang und der verwendete Sprachschatz deuten hingegen an, daß dieses Buch auf Computer der gehobenen Klasse zugeschnitten ist. Da Dinge wie Grafik, Ton oder Peripherieansteuerung sehr rechnerspezifisch sind, wird im Buch nicht darauf eingegangen.

Wer also eine gut zu lesende allgemeine und noch dazu umfangreiche Einführung ins Basic-Programmieren sucht, für den ist dieses Buch das Richtige. Sowohl als Lektüre zwischendurch wie bei der Arbeit am Computer bietet es sich an. Die sehr übersichtliche Gestaltung lädt direkt zum Nachschlagen ein, womit der Titel durchaus seine Berechtigung findet.

(Bernd Schulte)

Mitchell Waite/Miachel Pardee, Basic-Programmier-Handbuch, Markt&Technik 1984, 506 Seiten, ISBN 3-922120-92-X, 78 Mark

Das Maschinensprachebuch für Fortgeschrittene zum Commodore 64

Wer glaubt, daß das Thema »C 64« buchmäßig abgeschlossen sein müßte, wird von Data-Becker eines Besseren belehrt. Inzwischen sind über zwanzig Bücher dieses Verlags auf dem Markt. Eines davon ist das »Maschinensprachebuch für Fortgeschrittene zum Commodore 64«. Der Rückseitentext verspricht eine Einführung in die professionelle Maschinenspracheprogrammierung, angefangen bei der Problemanalyse, bis zu stets verwendbaren Tips und Tricks.

Wer jetzt erwartet, eine Art Lehrbuch oder Kursus zu bekommen, wird allerdings enttäuscht. Denn der tatsächliche Inhalt läßt sich grob in drei Teilgebiete aufteilen:

— Fließkommaarithmetik auf dem C 64 unter Ausnutzung schon vorhandener ROM-Routinen
— Interruptprogrammierung
— Selbstprogrammierte Basic-Erweiterungen

Diese drei Teilgebiete werden auf je zirka 70 Seiten ausführlich und mit sehr vielen Beispielen besprochen.

Dabei ist das Buch sehr verständlich geschrieben. So lernt man, wie versprochen, beim Lesen der einzelnen Kapitel und Ausprobieren der Beispiele, nicht nur etwas über die entsprechende Thematik, sondern auch über Maschinensprache im allgemeinen.

Nur ein Detail hat mir an diesem Buch mißfallen: Die ständige »Werbung« für den Data-Becker-Assembler Profimat.

Insgesamt kann ich das Buch folgenden zwei Gruppen wärmstens empfehlen:

Denen, die ihren C 64 besser kennenlernen wollen und denen ein ROM-Listing dazu nicht reicht, sowie denen, die gerne in Maschinensprache »weiterkommen« wollen.

(Boris Schneider)

Lothar Englisch, Maschinensprache für Fortgeschrittene, Data Becker 1984, 200 Seiten, 39 Mark

PEEKs und POKEs zum Commodore 64

Die große Reihe der Data-Becker-Bücher zum Commodore 64 ist um ein Buch reicher geworden. Es handelt sich um »PEEKs & POKEs zum Commodore 64« des Autors Hans Joachim Liesert.

Dieses Buch erweckte sofort mein Interesse, versprach doch der Titel endlich einmal ein Werk, mit dessen Hilfe man sich durch den »POKE-Wald« des Commodore 64 kämpfen konnte, ohne dabei an den Rand des Wahnsinns zu gelangen.

Dieses Buch ist für Anfänger gedacht und beginnt mit einer lockeren Einführung in die Arbeitsweise des Mikroprozessors 6502 und schließlich des gesamten Computers. Da die Informationen sehr ausführlich und genau nahegebracht werden, wird auch der blutigste Einsteiger schon innerhalb kürzester Zeit das Konzept eines Mikrocomputers verstehen.

Nach den Grundlagen beginnt der Autor die wichtigsten Adressen des Speichers zu erläutern und die Funktionsweise anhand von Beispielen ausführlich darzustellen. Das fängt bei der Peripherieverwaltung an, geht über die Grafik und den Ton, bis hin zur Tastatur und schließlich zu Basic und Betriebssystem.

Da praktisch keine Vorkenntnisse verlangt werden und das Buch zudem sehr erfrischend und spannend geschrieben ist, wird auch der Anfänger bei der Lektüre nicht überfordert, und er wird schnell mit den Möglichkeiten seines C 64 vertraut.

Etwas negativ bewerte ich nur den »Minikurs« für Maschinensprache am Ende des Buches, der eigentlich überflüssig ist, da sich die Adressaten des Buches, die noch ihre Anfangsschwierigkeiten mit PEEK und POKE überwinden müssen, sicherlich nicht in der Lage sehen, auch schon in Maschinensprache einzusteigen.

Insgesamt sicher ein empfehlenswertes Buch, bei dem auch der wichtige Speicherbelegungsplan zum C 64 nicht fehlt.

(Karsten Schramm)

Liesert, Peeks & Pokes zum Commodore 64, Data Becker 1984, 150 Seiten, 29 Mark



Auf das »!« kommt es an

Das folgende Programm wurde aus der Not geboren. Es erleichtert das Laden von Diskette und macht das umständliche Laden und Listen des Directory überflüssig. Nebenher lernen Sie eine Reihe nützlicher Maschinenroutinen kennen.

Geht es Ihnen auch so: Ich weiß nie genau, ob das Programm, das ich laden willl, nun »Disk Copy V 1.0« oder »Disk Copy V1.0« heißt. Versuche ich es mit »Disk*«, lade ich mit Sicherheit »Disk Monitor«.

Es hilft also nichts: Ich lade das Directory, liste es und — ärgere mich, weil das Programm, das ich laden wollte, ganz oben steht und beim Scrollen verschwindet. Also nochmal »LIST«, dann mit dem Cursor in die richtige Reihe fahren, »LOAD« eingeben, Cursor hinter den Programmnamen, »,8« eintippen. Der Computer antwortet mit einem verächtlichen »SYNTAX ERROR«, weil ich den Doppelpunkt mal wieder vergessen habe...

Vielleicht stelle ich mich besonders dumm an, aber es sollte auch anders gehen. Für Tätigkeiten, die meine geistigen Fähigkeiten übersteigen, gibt es doch Computer! Aus diesem Gedanken entstand das Programm.

Es wird mit »LOAD"!",8,1« geladen, startet automatisch, liest das Directory ein und schreibt hinter jedes Programm eine Nummer. Wenn der Bildschirm voll ist, wartet es auf meine Eingabe. Tippe ich »+«, bekomme ich die nächsten Programme angezeigt. Tippe ich aber die Programmnummer und dahinter ein »L«, wird das entsprechende Programm automatisch geladen. Handelt es sich um ein Maschinenprogramm, das absolut geladen werden muß (mit »,8,1«), tippe ich hinter die Nummer ein »A«.

So funktioniert es

Da ich nicht auf einen Autostart verzichten wollte, mußte das Programm vollständig in Maschinensprache geschrieben werden. Außerdem mußte es sehr kurz sein, denn ich hatte nur den Bereich von Speicherstelle 256 bis 600 ($0100 — $0258) zur Verfügung. Zu allem Unglück benötigen die Betriebssystemroutinen, die das Programm verwendet, auch noch Platz in diesem Bereich (Prozessorstack). Deshalb mußte ich auf alles verzichten, was nicht unbedingt nötig war. Es gibt also kein Menü für den Benutzer, ebenso keine Möglichkeit, das Programm zu unterbrechen, wenn man nichts laden will. Man kann nur »0L« eingeben, dann springt es mit einer Fehlermeldung ins Basic zurück. Trotzdem ist der verfügbare Speicher bis auf das letzte Byte belegt.

Die folgende Programmbeschreibung ist zwangsläufig sehr theoretisch. Das Programm läuft ja aber auch, wenn man sie nicht versteht. Machen Sie sich aber ruhig einmal die Mühe. Vielleicht können Sie doch den einen oder anderen nützlichen Tip bekommen. Wem es zu kompliziert wird, der kann bei den »Hinweisen zum Abtippen« weiterlesen.

Nehmen Sie also das Assembler-Listing zur Hand und folgen Sie mir in die Tiefen der Maschinensprache... Zur besseren Übersicht habe ich die einzelnen Teile beschrieben und immer die entsprechenden Zeilennummern dahintergesetzt.

Im Programmkopf bis Zeile 320 werden die Betriebssystemroutinen und die Variablen definiert. Sie sind nochmals im Kasten erläutert. Als Startadresse wird $010A festgelegt, der Maschinencode kann aber nicht direkt dorthin assembliert

```
10 FORI=49162TO49495:READA:POKEI,A:S=S+A
20 NEXT:IFS<>35775THENPRINT"FEHLER IN DA
TAS":STOP
30 PRINT"{CLR}BITTE LEGEN SIE EINE NEUE DISK
ETTE EIN. {DOWN}DISKETTE WIRD FORMATIERT !"
40 GETA$:IFA$=""THEN40
50 OPEN1,8,15,"N0:MENU,MN":CLOSE1
60 PRINT"{CLR}{3 DOWN}POKE43,0:POKE44,192:POKE45,
90:POKE46,193"
70 PRINT"{2 DOWN}SAVE"CHR$(34)"!"CHR$(34)",8"
80 PRINT"{HOME}";
90 FORI=631TO641:POKEI,13:NEXT:POKE198,1
0
100 DATA165,43,133,57,165,44,133,58,169,
13,32,210,255,160,0,177,57,133,59,200
110 DATA177,57,133,60,200,177,57,170,200
,177,57,32,205,189,160,3,200,177,57
120 DATA240,5,32,210,255,208,246,165,59,
133,57,165,60,133,58,160,1,177,57,240
130 DATA23,169,27,133,211,169,0,166,251,
32,205,189,230,251,230,253,165,253
140 DATA201,21,144,183,169,128,133,254,1
69,13,32,210,255,162,0,32,207,255,157
150 DATA64,3,232,201,13,208,245,202,202,
189,64,3,201,43,208,12,169,0,133,253
160 DATA36,254,48,144,133,251,16,132,201
,76,208,6,169,0,133,252,240,8,201,65
170 DATA208,208,169,128,133,252,202,189,
64,3,41,15,133,251,202,48,16,189,64
180 DATA3,41,15,170,24,165,251,105,10,13
3,251,202,208,247,166,251,24,165,43
190 DATA105,35,133,57,165,44,133,58,202,
240,111,24,165,57,105,32,133,57,144
200 DATA244,230,58,176,240,36,2,2,2,2,2,
2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2
210 DATA2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,
2,2,2,2,2,2,2,2,2,2,2,169,1,162,208
220 DATA160,1,32,189,255,162,8,160,0,32,
186,255,152,133,253,133,251,166,43
230 DATA164,44,32,213,255,32,51,165,76,1
0,1,160,0,177,57,201,34,240,3,200,208
240 DATA247,152,96,32,37,2,56,101,57,133
,57,32,37,2,166,57,164,58,32,189,255
250 DATA169,1,162,8,160,0,132,10,36,252,
16,2,160,1,32,186,255,76,111,225
READY.
```

Listing »Lader«

werden, sonst würde ja der Autostart anlaufen. Deshalb wird er zunächst in den Bereich $C00A — $C158 geschrieben.

Das Programm selbst beginnt mit Zeile 1510. Die Routine »DIRIN« lädt das Directory in den Speicher. Dazu benutzen wir die Betriebssystemroutine »LOAD«. Diese braucht zur Vorbereitung die Routinen SETLFS und SETNAM. SETLFS bestimmt die logische Filenmmer (01), die Geräteadresse (08) und die Sekundäradresse (0 für Lesen). SETNAM übergibt den Filenamen, in unserem Falle »$«. LOAD muß schließlich noch wissen, wohin das Directory geladen werden soll. Diese Adresse (Anfang des Basic-Speichers, steht in ($2B/$2C) wird im X- und Y-Register übergeben. Um die richtigen Koppeladressen zu erhalten, muß anschließend noch die Basic-Routine aufgerufen werden, die das für uns erledigt ($A533). Jetzt steht unser Directory genauso im Speicher wie sonst und wartet darauf, auf dem Bildschirm ausgegeben zu werden.



Alle Programme sind durchnumeriert

PRUEFSUMMENLISTE
BLOCKGROESSE 20

ZEILE	ANZAHL	SUMME	KEIN POKE?
100	20	2263	
120	40	4991	
130	60	7435	
140	80	10735	
150	100	13594	
160	120	16221	
170	140	19025	
180	160	21080	
190	180	23840	
200	200	26287	
200	220	26327	
210	240	26367	
220	260	27732	
230	280	30452	
240	300	32441	
250	320	34281	
GESAMT	334	35775	

Prüfsummen »Lader«

Die Ausgabe kann aber nicht wie beim normalen Listen erfolgen, denn wir wollen ja hinter jedem Namen eine Nummer ausgeben. Also müssen wir das selbst in die Hand nehmen. Mit einem kühnen Sprung geht es deshalb jetzt nach DIROUT. Sehen wir uns nun den Speicherausdruck eines Directory etwas genauer an:

Es geht los bei $0801. Dort steht die Adresse des nächsten Eintrages $081F. Wie immer steht im Speicher zuerst das Low-Byte und dahinter das High-Byte. Es folgt der Diskettenname, der in diesem Falle mit dem des Verfassers auffallend übereinstimmt. Der erste Eintrag beginnt also mit Adresse $081F. Dort finden wir wieder die Adresse des nächsten Eintrages ($083F). Jeder Eintrag hat übrigens die gleiche Länge, nämlich 32 Byte. Wir werden davon später noch Gebrauch machen.

Die 2 Byte hinter der Koppeladresse enthalten die Länge des Programms in Blöcken, in unserem Falle also $0002. Nun kommen ein paar Leerzeichen ($20). Wieviele, hängt davon ab, ob die Blocklänge ein, zwei oder drei Ziffern benötigt. Die Namen sollen ja beim Listen schließlich ordentlich untereinander stehen. Jetzt folgt der Filename, von Anführungszeichen eingeschlossen. Sie haben es sicher schon bemerkt: Das Directory dieser Diskette hat als erstes Programm das »!« wie sich das für anständige Disketten gehört!

Jetzt wieder einige Leerzeichen und schon ist der nächste Fileeintrag erreicht. Das geht so weiter bis eine Koppeladresse erreicht wird ($089D in Adresse $087F), die aus dem Rahmen fällt. In Adresse $089D stehen nämlich zwei jämmerliche Nullen. So, sagt sich das Betriebssystem, jetzt ist aber Schluß. Und es hat wie so oft recht...

Zurück zu unserem Programm. Es soll ja jetzt ein Directory auf den Bildschirm ausgeben, und zwar schöner, als es das Betriebssystem jemals könnte. Wir initialisieren also unsere Variable USE mit dem Basic-Anfang ($0801). In Zeile 390 beginnt eine Schleife, die immer eine Zeile ausgibt. Zuerst holen wir uns die Koppeladresse und speichern sie in NEXT (Zeile 410 — 460). Als nächstes müssen wir die 2 Byte mit der Blocklänge in eine ordentliche Dezimalzahl umwandeln. Das erledigt für uns die Basic-Routine »NUMOUT« (470 — 520). Jetzt brauchen wir nur noch den Namen mit allen Leerzeichen davor und dahinter, auszugeben (530 — 580).

Eigentlich müßte nun die Programmnummer ausgegeben werden, aber damit warten wir noch. Aus folgendem Grund: Wir wollen diese Schleife ja für jede Zeile benützen, auch für die letzte. In der steht aber die »BLOCKS FREE«-Meldung und dahinter darf ja keine Nummer mehr erscheinen. Wir müssen also vorher prüfen, ob wir schon die letzte Zeile erreicht haben.

Dazu übertragen wir die Adresse des nächsten Files, die wir in NEXT aufbewahrt haben, nach USE. Zeigt USE jetzt auf eine Null, sind wir fertig und springen ans Ende unserer Routine, nach DIR4 (590 — 650). Ist das aber noch nicht der Fall, geben wir unsere Programmnummer aus. Den Cursor stellen wir dazu auf Spalte 27 ($1B) und benutzen wieder NUMOUT, um FILENR als Dezimalzahl auszugeben (660 — 700). Eine Zeile wäre geschafft! Im folgenden bereiten wir uns auf die nächste Filenummer vor und springen wieder nach oben zu DIR1 zurück. Sollten aber schon 20 Zeilen auf dem Bildschirm stehen, merken wir uns im ENDFLG, daß wir noch nicht fertig sind und verlassen die Ausgaberoutine vorerst. Erinnern Sie sich, wir wollten vermeiden, daß der Bildschirm ohne unseren ausdrücklichen Befehl wegscrollt (710 — 800).

Der Bildschirm hat sich inzwischen gefüllt, unsere Nummern stehen fein säuberlich hinter den Programmen, nun sollte uns ein freundlich blinkender Cursor dazu animieren, dem Computer mitzuteilen, wie es weitergehen soll. Also hinein in die Routine WAHL.

Das Wichtigste erledigt hier die Betriebssystemroutine CHRIN. Sie läßt den Cursor blinken, bis »RETURN« gedrückt wird, schreibt die eingegebenen Zeichen auf den Bildschirm und liest sie anschließend vom Bildschirm wieder ab, damit wir sie schön der Reihe nach bearbeiten können. Wir speichern alle Zeichen zunächst im BUFFER und sehen uns dann das letzte Zeichen genauer an (820 — 900). Als erstes prüfen wir, ob ein »+« eingegeben wurde. Wenn ja, setzen wir FILEANZ wieder auf Null, und, je nachdem ob das ENDFLG Ende signalisiert oder nicht, auch die FILENR. Anschließend erfolgt der Rücksprung in die Directory-Ausgabe (910 — 980).

```
STARTADRESSE

10 OPEN1,8,15
20 OPEN2,8,2,"#"
30 PRINT#1,"U1 2 0 17 0"
40 PRINT#1,"B-P 2 3"
50 PRINT#2,CHR$(1)
60 PRINT#1,"U2 2 0 17 0"
70 CLOSE2:CLOSE1
READY.
```

Ändern der Startadresse. Lesen Sie hierzu die »Hinweise zum Abtippen«.

Wurde kein »+« eingegeben, prüfen wir weiter auf »L« beziehungsweise »A«. In Abhängigkeit davon setzen wir das ABSFLG (990 — 1070). Jetzt müssen wir herausfinden, welche Programmnummer geladen werden soll. Dazu müssen wir die ein oder zwei Dezimalzahlen in ein Hexbyte umwandeln, weil unser Computer nun mal nichts anderes versteht. Andererseits ist es ja nicht einzusehen, daß wir uns auf sein mathematisches Niveau herabbegeben und unsere Zahlen demnächst als Hex- oder noch schlimmer als Binärzahlen eingeben. Das Umwandeln ist ja gar nicht so schwierig. Wir holen uns die Einerziffer und zählen dann so oft 10 ($0A) dazu, wie die Zehnerziffer angibt. Da der Computer sich unsere Ziffern aber nicht als 0,1...9 merkt, sondern als $30, $31... $3A, müssen wir jeweils die oberen 4 Bits ausmaskieren. Das war's denn auch schon. In FILENR steht zur Belohnung tatsächlich die gewünschte Filenummer mundgerecht für unseren Computer (1080 — 1200).

Jetzt haben wir das Schlimmste — fast — hinter uns. Wir brauchen nur noch den passenden Filenamen zu suchen. Den holen wir uns aus dem Directory. In bewährter Weise benutzen wir unsere Variable USE. Um den Diskettennamen zu überlesen, zählen wir zum Basic-Anfang in $2B/$2C 35 ($23) dazu. Damit liegen wir so ungefähr richtig, auf jeden Fall vor dem ersten Namen. Erinnern sie sich, jeder Eintrag im Directory belegt genau 32 Byte. Allerdings beginnen die Filenamen nicht immer so schön regelmäßig wie in unserem Beispiel, immer an der selben Stelle. Wir müssen den Anfang des Namens also noch genau suchen. Vorher brauchen wir aber erst einmal die richtige Stelle im Directory. Wir addieren zu USE jetzt daher so oft 32 ($20), wie in FILENR angegeben (1250 — 1400).

Jetzt müssen wir in unserem Programm ein Stück überspringen, weil die folgenden Bytes wegen des Autostarts auf 2 gesetzt werden müssen. Weiter geht es mit LAD2 in Zeile 1780. Vorher steht allerdings im Programm noch die Subroutine FINDA.E, was so viel wie »Finde den Anfang beziehungsweise das Ende des Namens« bedeutet. Sie durchsucht den Text, auf den USE zeigt, nach einem Anführungszeichen und übergibt im Akku, wieviel Zeichen es bis dahin sind (1680 — 1750).

Diese Zahl addieren wir zu USE — eins mehr, denn das Anführungszeichen selbst gehört ja nicht mit zum Namen. Mit FINDA.E erhalten wir im Akku die Länge des Namens, so wie es die Routine SETNAM, die wir oben schon benutzt haben, verlangt. Noch ein kurzer Sprung nach SETLFS, wo wir in Abhängigkeit vom ABSFLG als Sekundäradresse Null oder Eins übergeben. Den Rest erledigt die Basic-Routine BASICLOAD (1780 — 1940). Geschafft, die schönste Zeile im Assemblerlisting ist erreicht: .EN heißt ENDE.

Sicher, es war nicht ganz einfach, aber wenn Sie mir bis hierhin gefolgt sind, haben Sie eine ganze Reihe kleiner Routinen gelernt, wie sie in jedem Maschinenprogramm gebraucht werden.

NUMOUT ($BDCD) — wandelt zwei Hexbytes in eine Dezimalzahl um und gibt sie aus. (HByte im Akku, LByte im X-Reg)
SETLFS ($FFBA) — übergibt logische Filenummer (Akku), Geräteadresse (X-Reg) und Sekundäradresse (Y-Reg).
SETNAM ($FFBD) — übergibt den Filenamen. Akku enthält die Länge des Namens, X-Reg das LByte, Y-Reg das HByte der Adresse, an der der Name beginnt.
LOAD ($FFD5) — lädt das durch SETLFS und SETNAM bezeichnete File an die Adresse, die im X-Reg (LByte) und im Y-Reg (HByte) angegeben ist. Der Akku enthält 0 für LOAD und 1 für VERIFY.
CHRIN ($FFCF) — holt Zeichen von dem in $99 festgelegten Eingabegerät. Der Cursor blinkt, bis RETURN eingegeben wird. Die Zeichen werden dann vom Bildschirm gelesen und im Akku übergeben.
CHROUT ($FFD2) — gibt das ASCII-Zeichen im Akku aus.
Variable
BUFFER — ist ein Zwischenspeicher.
USE und NEXT — enthalten die Adresse des gegenwärtigen beziehungsweise nächsten Directory-Eintrages.
FILENR — ist die aktuelle Filenummer im Directory.
ABSFLG — ist ein Flag, das normales oder absolutes Laden signalisiert.
FILEANZ — enthält die Anzahl der auf dem Bildschirm ausgedruckten Files.
ENDFLG — ist ein Flag, das signalisiert, ob das Ende des Directory bereits erreicht ist.

Betriebssystem-Routinen

Hinweise zum Abtippen

Das Programm kann nicht an die Stelle geschrieben werden, an der es endgültig stehen soll, weil sonst ja der Autostart anlaufen würde. Wir schreiben es daher zuerst nach 49152 ($C000). Tippen Sie zuerst den Basic-Lader ein. Nach RUN muß der Text aus Zeile 30 erscheinen, sonst haben Sie einen Fehler gemacht. Legen Sie nun eine neue Diskette ein, und drücken Sie eine Taste. Das Programm wird nun auf die Diskette geschrieben, allerdings mit der falschen Startadresse. Schalten Sie danach bitte den Computer aus und wieder ein. Um die falsche Startadresse zu ändern, geben Sie das Programm »Startadresse« ein und starten Sie es mit RUN. Jetzt wird die Adresse geändert, und Sie haben das »!« lauffähig auf Ihrer Diskette. Um es auf andere Disketten zu übertragen, verwenden Sie ein Kopierprogramm, zum Beispiel »Super Copy«.

Und dann lehnen Sie sich zurück, geben LOAD »!«,8,1 ein und genießen die Früchte Ihrer Arbeit.

(Dietrich Weineck/rg)

```
0020;*********************
0030;*                   *
0040;* LADEPROGRAMM !    *
0050;* MAI 1984          *
0060;* N.MANN & D.WEINECK *
0070;* FLEETRADE 40      *
0080;* 2800 BREMEN       *
0090;* TEL. 0421 / 493090 *
0100;*                   *
0110;*********************
0120;
0130;
0140NUMOUT      .DE $BDCD
0150BASICLOAD   .DE $E16F
0160SETLFS      .DE $FFBA
0170SETNAM      .DE $FFBD
0180LOAD        .DE $FFD5
0190CHRIN       .DE $FFCF
0200CHROUT      .DE $FFD2
0210;
0220BUFFER      .DE $0340
0230USE         .DE $39
0240NEXT        .DE $3B
0250FILENR      .DE $FB
0260ABSFLG      .DE $FC
0270FILEANZ     .DE $FD
0280ENDFLG      .DE $FE
0290;
0300            .BA $010A
0310            .MC $C00A
0320            .OS
0330;
0340;
0350DIROUT      LDA *$2B
0360            STA *USE
0370            LDA *$2C
0380            STA *USE+1
0390DIR1        LDA #$0D
0400            JSR CHROUT
0410            LDY #$00
0420            LDA (USE),Y
0430            STA *NEXT
0440            INY
0450            LDA (USE),Y
0460            STA *NEXT+1
0470            INY
0480            LDA (USE),Y
0490            TAX
0500            INY
0510            LDA (USE),Y
0520            JSR NUMOUT
0530            LDY #$03
0540DIR2        INY
0550            LDA (USE),Y
0560            BEQ DIR3
0570            JSR CHROUT
0580            BNE DIR2
0590DIR3        LDA *NEXT
0600            STA *USE
0610            LDA *NEXT+1
0620            STA *USE+1
0630            LDY #$01
0640            LDA (USE),Y
0650            BEQ DIR4
0660            LDA #$1B
0670            STA *$D3
0680            LDA #$00
0690            LDX *FILENR
0700            JSR NUMOUT
0710            INC *FILENR
0720            INC *FILEANZ
0730            LDA *FILEANZ
0740            CMP #$15
0750            BCC DIR1
0760            LDA #$80
0770DIR4        STA *ENDFLG
0780            LDA #$0D
0790            JSR CHROUT
0800;
0810;
0820WAHL        LDX #$00
0830WAHL1       JSR CHRIN
0840            STA BUFFER,X
0850            INX
0860            CMP #$0D
0870            BNE WAHL1
0880            DEX
0890            DEX
0900            LDA BUFFER,X
0910            CMP #'+'
0920            BNE WAHL2
0930            LDA #$00
0940            STA *FILEANZ
0950            BIT *ENDFLG
0960            BMI DIR1
0970            STA *FILENR
0980            BPL DIROUT
0990WAHL2       CMP #'L'
1000            BNE WAHL3
1010            LDA #$00
1020            STA *ABSFLG
1030            BEQ WAHL4
1040WAHL3       CMP #'A'
1050            BNE WAHL
1060            LDA #$80
1070            STA *ABSFLG
1080WAHL4       DEX
1090            LDA BUFFER,X
1100            AND #$0F
1110            STA *FILENR
1120            DEX
1130            BMI LADEN
1140            LDA BUFFER,X
1150            AND #$0F
1160            TAX
1170            CLC
1180WAHL5       LDA *FILENR
1190            ADC #$0A
1200            STA *FILENR
1210            DEX
1220            BNE WAHL5
1230;
1240;
1250LADEN       LDX *FILENR
1260            CLC
1270            LDA *$2B
1280            ADC #$23
1290            STA *USE
1300            LDA *$2C
1310            STA *USE+1
1320LAD1        DEX
1330            BEQ LAD2
1340            CLC
1350            LDA *USE
1360            ADC #$20
1370            STA *USE
1380            BCC LAD1
1390            INC *USE+1
1400            BCS LAD1
1410;
1420;
1430TEXT1       .BY '$'
1440;
1450;
1460            .BY 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
1470            .BY 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
1480            .BY 2 2 2 2 2 2 2 2 2 2 2 2 2 2
1490;
1500;
1510DIRIN       LDA #$01
1520            LDX #L,TEXT1
1530            LDY #H,TEXT1
1540            JSR SETNAM
1550            LDX #$08
1560            LDY #$00
1570            JSR SETLFS
1580            TYA
1590            STA *FILEANZ
1600            STA *FILENR
1610            LDX *$2B
1620            LDY *$2C
1630            JSR LOAD
1640            JSR $A533
1650            JMP DIROUT
1660;
1670;
1680FINDA.E     LDY #$00
1690CMP1        LDA (USE),Y
1700            CMP #$22
1710            BEQ CMP2
1720            INY
1730            BNE CMP1
1740CMP2        TYA
1750            RTS
1760;
1770;
1780LAD2        JSR FINDA.E
1790            SEC
1800            ADC *USE
1810            STA *USE
1820            JSR FINDA.E
1830            LDX *USE
1840            LDY *USE+1
1850            JSR SETNAM
1860            LDA #$01
1870            LDX #$08
1880            LDY #$00
1890            STY *$0A
1900            BIT *ABSFLG
1910            BPL LAD3
1920            LDY #$01
1930LAD3        JSR SETLFS
1940            JMP BASICLOAD
1950            .EN
```

Das Assemblerlisting zum Ladeprogramm

Spielen in der dritten Dimension

»3D-Vier gewinnt« ist eine interessante Variante des bekannten Strategiespiels. Bemerkenswert ist auch, daß der Autor ohne Steuerzeichen ausgekommen ist.

Bei »3D-Vier gewinnt« setzen die beiden Spieler abwechselnd einen Spielstein auf eines der 16 Felder, die in vier Reihen und vier Spalten angeordnet sind. Hierbei werden Türme von maximal vier Steinen Höhe gebildet. Sieger ist, wer als erster eine beliebige Reihe oder Diagonale — auch Raumdiagonale — mit vier Steinen besetzt. Sollten zuvor alle 64 möglichen Felder besetzt sein, so geht das Spiel unentschieden aus.

Zum Programm:

Man kann wahlweise gegen einen Spielpartner oder, bei unterschiedlicher Spielstärke, gegen den Computer antreten. Der Computer benötigt hierbei, obwohl das Programm in reinem Basic geschrieben ist, weniger als zehn Sekunden Bedenkzeit. Es ist auch möglich, den Computer sich selbst zu überlassen. Das Programm übernimmt die dreidimensionale Darstellung, überwacht die Korrektheit der Züge und ermittelt den Sieger.

Während des Spieles kann man sich jederzeit vom Computer einen Zugvorschlag holen, der allerdings mit Vorsicht zu genießen ist (da Spielstärke 1 voreingestellt ist) und auf Wunsch die Seiten wechseln. Alle nötigen Eingaben werden im Dialog erfragt. Bleibt nur zu erklären, wie man einen Zug ausführt: Während des Spieles ist in der linken Bildschirmhälfte die dreidimensionale Darstellung des Spielfeldes zu sehen. Rechts erscheint in schematisierter Form eine Draufsicht auf das Spielfeld, wobei in deren unteren linken Ecke ein Cursor erscheint. Dieser läßt sich nun, entweder mit einem Joystick in Port 2 oder mittels der Cursortasten an die gewünschte Position dirigieren. Drückt man nun den Feuerknopf beziehungsweise die Return-Taste, wird der gewünschte Zug — sofern er den Regeln entspricht — ausgeführt.

Bei Spielende ertönt eine kleine Melodie, und das Programm verdeutlicht durch Blinken der entsprechenden Spielsteine die Gewinnsituation. Außerdem wird der Name des Siegers angezeigt. Durch Drücken einer beliebigen Taste startet man ein neues Spiel.

Detailbeschreibung — Spielstrategie:

Jedes Feld ist durch eine x-, y- und z-Koordinate eindeutig bestimmt. Damit der Computer etwas zu rechnen hat, wird jedem möglichen Zustand der einzelnen Felder ein Zahlenwert zugeordnet, der im dreidimensionalen Feld FE gespeichert wird. Hierbei bedeutet

3 = das Feld ist von Spieler 2 besetzt
2 = das Feld ist von Spieler 1 besetzt
1 = das Feld ist unbesetzt, kann aber im nächsten Zug besetzt werden
0 = unbesetztes Feld

Ist der Computer am Zug oder wird ein Zugvorschlag gewünscht, geschieht folgendes: Der Computer berechnet für jedes mögliche Feld (FE(X,Y,Z)=1) eine Bewertung (im Programm ab Zeile 2000) und entscheidet sich dann für das Feld mit der höchsten Bewertung. Mit einem kleinen Trick wurde der Zeitaufwand hierfür minimiert: Für die Entscheidung, ob ich in die linke untere Ecke setze, ist es unwesentlich, wie es rechts oben aussieht. Das Programm bewertet also jede Viererreihe getrennt und ermittelt die Bewertung eines Feldes als Summe der Bewertungen aller Viererreihen, an denen dieses Feld beteiligt ist. Der große Zeitvorteil ergibt sich jetzt dadurch, daß nach jedem Zug nur einfach jene Reihen neu bewertet werden müssen, in denen sich wirklich etwas geändert hat (ab Programmzeile 2500).

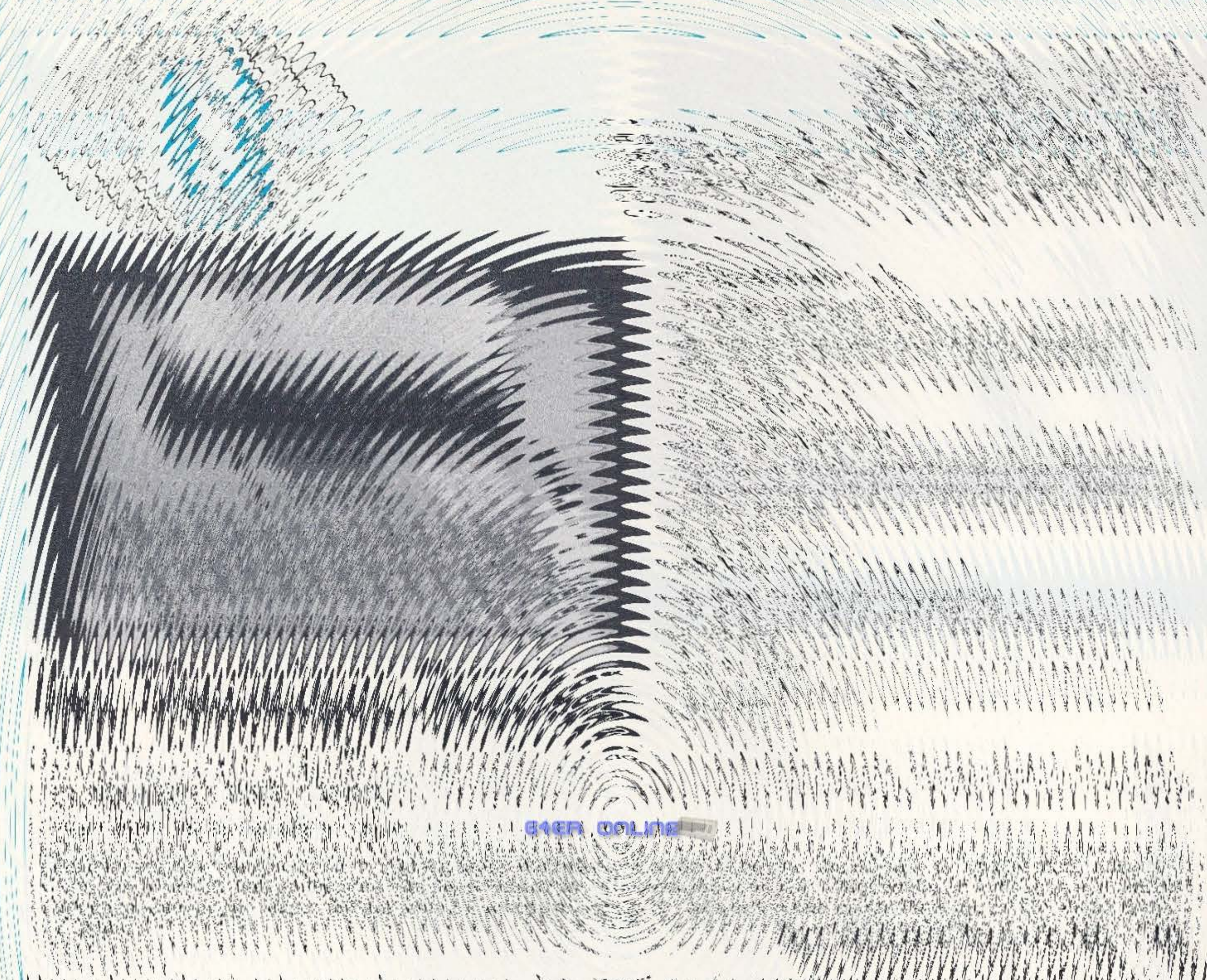
Bei der Spielstärke 2 wird im Gegensatz zur Spielstärke 1 mit berücksichtigt, daß das darüberliegende Feld im nächsten Zug vom Gegner besetzt werden kann. Deshalb wird dieses Feld (mit negativem Vorzeichen) ebenfalls bewertet. Dadurch verdoppelt sich allerdings die Bedenkzeit bei Spielstärke 2.

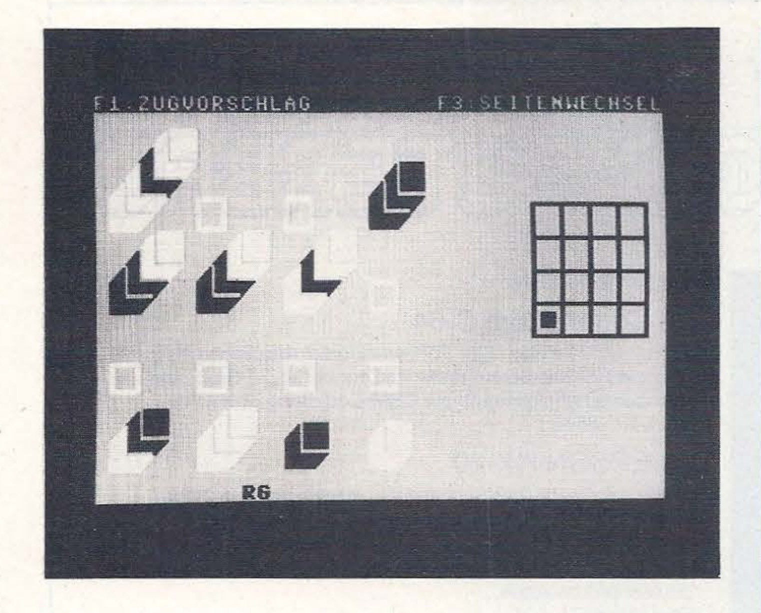
In den Programmzeilen 3010, 3100 und 3180 steht der Aufruf »SYS 58732«. Hierbei handelt es sich um eine Betriebssystem-Routine, die nach der Cursorpositionierung (durch Setzen der Speicherstellen 211 und 214) die Bildschirmparameter neu bestimmt.

(Uwe Weiß/rg)

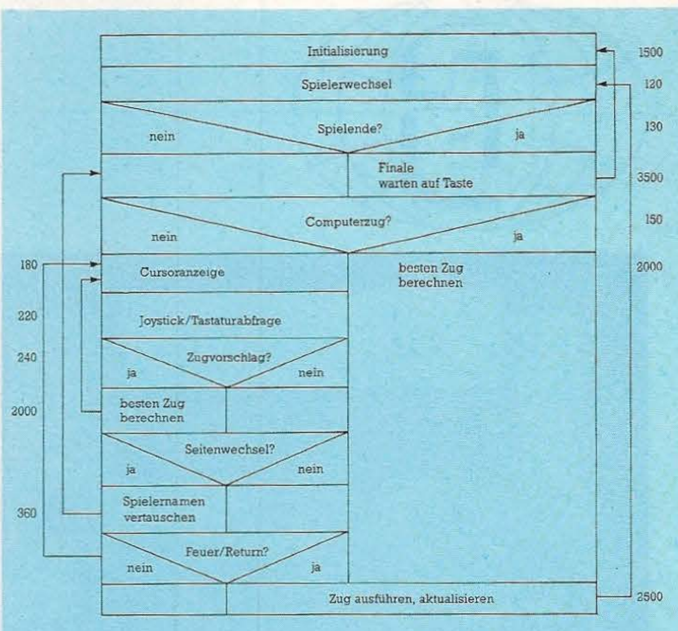
Variablenliste von »3D-Vier gewinnt«

ZG	: Zahl der durchgeführten Züge
PL	: aktueller Spieler
GW	: Gewinnsituation
X,Y,Z	: Zugkoordinaten
X0,Y0	: Bildschirmkoordinaten
D1,D2,D3,D4	: Bewertung Raumdiagonalen
BL$	: 40 Leerzeichen (Blanks)
Feldvariablen:	
FE	: Speicherung des gesamten Spielfeldes
BE	: Höhe der Türme/besetzte Felder
BW	: Gesamtbewertung der Felder
RX,RY,RZ	: Bewertung Viererreihe in x,y,z-Richtung
OX,OY,OZ,UY,UZ	: Bewertung Flächendiagonalen
W$,X$	: 3dim. Darstellung der Spielsteine
SP$	: Spielernamen
CO$	: Farbe der Steine
CL$	: dto. für Blinken bei Spielende
SS	: Spielstärke





So sieht's auf den Bildschirm aus



Schematische Darstellung von »3D-Vier gewinnt«

Listing »3D-Vier gewinnt«

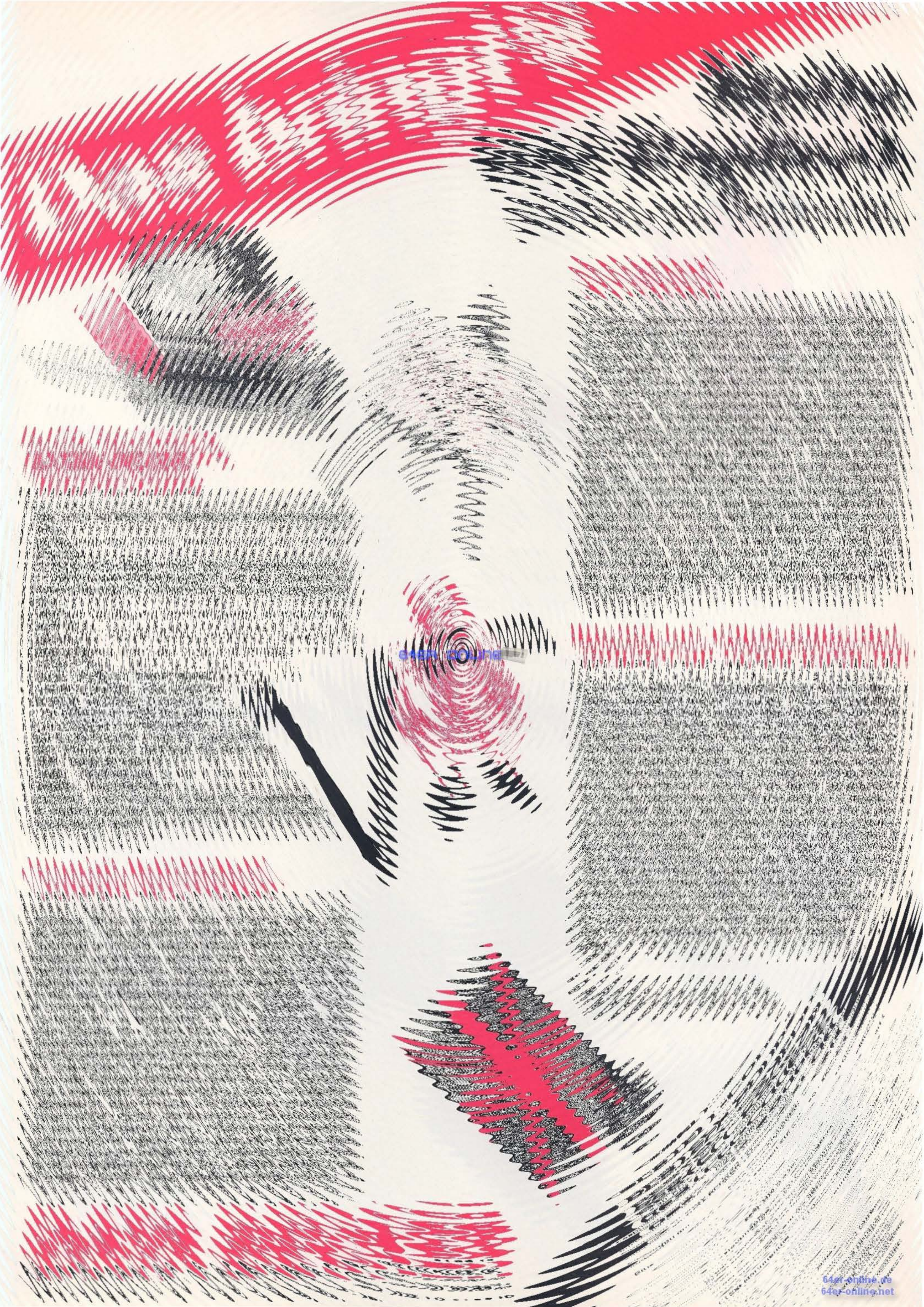
```
10 REM        ******************
11 REM        *                *
12 REM        * 3D-VIER-GEWINNT *
13 REM        *                *
14 REM        *      1984 BY   *
15 REM        *                *
16 REM        *  UWE WEISS     *
17 REM        *  LOESKENWEG 60 *
18 REM        *  4300   ESSEN 1 *
19 REM        *                *
20 REM        * TEL 0201/326366 *
21 REM        *                *
22 REM        ******************
23 :
24 :
25 :
26 :
27 :
28 :
29 :
30 :
100 GOSUB1500:REM INITIALISIERUNG
110 ZG=0:PL=3:GW=0
120 PL=5-PL:ZG=ZG+1
130 IF(GW<>0)OR(ZG>64)THEN3500:REM SPIEL
ENDE
140 PRINTCO$(PL):GOSUB3100
141 H=(20-LEN(SP$(PL)))/2:X0=1:Y0=24
145 X$=LEFT$(BL$,H)+SP$(PL)+LEFT$(BL$,10
):GOSUB3010:PRINTCHR$(19)
150 IFSP$(PL)="C 64"THEN500
160 REM *** SPIELERZUG ***
170 X=0:Y=0:A0=1575
180 A1=1575+2*X-80*Y
200 POKEA0,32:A0=A1
210 POKEA1,160:POKEA1+54272,CO(PL)
220 J=PEEK(56320)AND31
230 GETK$:K=ASC(K$+CHR$(0))
235 IF(K=0)AND(J=31)THEN220
240 IFK=133THEN350:REM ZUGVORSCHLAG
250 IFK=134THEN360:REM SEITENWECHSEL
260 IF(K=13)OR((JAND16)=0)THEN320
270 IF(K=145)OR((JAND1)=0)THENIFY<3THENY
=Y+1
280 IF(K=17)OR((JAND2)=0)THENIFY>0THENY=
Y-1
290 IF(K=157)OR((JAND4)=0)THENIFX>0THENX
=X-1
300 IF(K=29)OR((JAND8)=0)THENIFX<3THENX=
X+1
310 GOTO180
320 IFBE(X,Y)=4THEN220
330 GOSUB2500:REM ZUG AUSFUEHREN
340 GOTO120
350 POKEA1,32:GOSUB2000:X=ZX:Y=ZY:GOTO18
0
360 SP$=SP$(2):SP$(2)=SP$(3):SP$(3)=SP$:
GOTO140
500 REM *** COMPUTERZUG ***
510 GOSUB2000
520 X=ZX:Y=ZY
530 AD=1575+2*X-80*Y
540 POKEAD,160:POKEAD+54272,CO(PL)
550 GOSUB2500:POKEAD,32
560 GOTO120
1500 IFRU=1THEN1645
1505 RU=1
1510 DIMFE(3,3,3),BE(3,3),BW(3,3),W(3,3)
1511 DIMRX(3,3),RY(3,3),RZ(3,3),MX(15)
1512 DIMUX(3),UY(3),UZ(3),OX(3),OY(3),OZ
(3)
1513 DIMX$(3),W$(3),SP$(3),CO$(3),CL$(3)
1514 DIMX(3),Y(3),Z(3),SS(3),CO(3)
1515 H0$=CHR$(18)+"▛"
1516 H1$=CHR$(17)+CHR$(157)
1517 H2$=H1$+CHR$(157)+CHR$(157)
1518 H3$=CHR$(146)+"▛"
1520 W$(0)=H0$+H1$+"| "+H1$+"└▁"+H3$
```

```
1525 W$(1)=H0$+H1$+" "+H1$+"  "+H3$
1530 W$(2)=H0$+"|  "+H2$+"|└─"+H2$+"└─"+H
3$
1535 W$(3)=H0$+"|  "+H2$+" └─"+H2$+"  "+H
3$
1536 X$(0)=W$(3)
1537 FORI=1TO3:X$(I)=W$(2):NEXT
1539 PRINTCHR$(147):POKE53280,0:POKE5328
1,12
1540 PRINTCHR$(19)CHR$(18)CHR$(144);
1545 BL$="
        "
1550 PRINTLEFT$(BL$,12)"3D-VIER-GEWINNT"
LEFT$(BL$,13)
1555 PRINT:PRINT:PRINT:PRINT" BITTE WAEH
LEN SIE:"
1560 PRINT:PRINT" 1 = SCHWARZWEISSFERNSE
HER"
1565 PRINT:PRINT" 2 = FARBFERNSEHER"
1570 GETK$:K=VAL(K$)
1575 ONKGOTO1585,1605
1580 GOTO1570
1585 CO$(2)=CHR$(5):CO$(3)=CHR$(144)
1590 CL$(2)=CHR$(155):CL$(3)=CHR$(151)
1595 CO(2)=1:CO(3)=0
1600 GOTO1620
1605 CO$(2)=CHR$(28):CO$(3)=CHR$(31)
1610 CL$(2)=CHR$(150):CL$(3)=CHR$(154)
1615 CO(2)=2:CO(3)=6
1620 SP$(2)="C 64":SP$(3)="C 64"
1625 S=54272:POKES+23,113:POKES+24,31
1630 POKES+2,0:POKES+3,8:POKES+4,0
1635 POKES+5,21:POKES+6,240
1640 GOTO1670
1645 SP$=SP$(2):SP$(2)=SP$(3):SP$(3)=SP$
1650 FORI=0TO3:FORJ=0TO3
1655 BE(I,J)=0
1660 FORK=1TO3:FE(I,J,K)=0
1665 NEXTK,J,I
1670 PRINTCHR$(147)CHR$(18)CHR$(144);
1675 PRINT"F1:ZUGVORSCHLAG";LEFT$(BL$,9)
;"F3:SEITENWECHSEL"
1680 PRINTCHR$(5)
1685 FORI=1TO4
1690 PRINT:PRINT:PRINT
1695 PRINT" ┌┐    ┌┐    ┌┐    ┌┐"
1700 PRINT" └┘    └┘    └┘    └┘":NEXT
1705 FORI=2TO3
1710 PRINTCHR$(19)
1715 PRINTCO$(I)"SPIELER";I-1;"  "SP$(I)
;BL$
1720 PRINTCHR$(145)CHR$(145);SPC(9)" ";
1725 INPUTSP$(I)
1730 SS(I)=1
1735 IFSP$(I)<>"C 64"THEN1755
1740 PRINT"SPIELSTAERKE 1/2"
1745 GETK$:IFK$="2"THENSS(I)=2:GOTO1755
1750 IFK$<>"1"THEN1745
1755 PRINTCHR$(19)
1760 PRINTLEFT$(BL$,30):PRINTLEFT$(BL$,3
0)
1765 NEXT
1770 FORI=0TO3:FORJ=0TO3
1775 RX(I,J)=1/16:RY(I,J)=1/16:RZ(I,J)=1
/8:FE(I,J,0)=1
1780 NEXT:NEXT
1785 FORI=0TO3
1790 RX(I,0)=1:RY(I,0)=1
1795 UX(I)=1/8:UY(I)=1/8:UZ(I)=1/16
1800 OX(I)=1/8:OY(I)=1/8:OZ(I)=1/16
1805 NEXT
1810 UZ(0)=1:OZ(0)=1
1815 D1=1/8:D2=D1:D3=D1:D4=D1
1820 RETURN
2000 XX=0:GOSUB2200
2010 IFSS(PL)=1THEN2100
2020 FORI=0TO3:FORJ=0TO3
2030 W(I,J)=BW(I,J)
2040 NEXT:NEXT
2050 XX=1:GOSUB2200
2060 FORI=0TO3:FORJ=0TO3
2070 BW=BW(I,J):BW(I,J)=W(I,J)
2080 IF(W(I,J)<64)AND(BW>0)THENBW(I,J)=B
W(I,J)-BW/2
2090 NEXT:NEXT
2100 MAX=-5000:H=0
2110 FORI=0TO3:FORJ=0TO3
2120 BW=BW(I,J)
2130 IFBW=MAXTHENMX(H)=10*I+J:H=H+1
2140 IFBW>MAXTHENH=1:MX(0)=10*I+J:MAX=BW
2150 NEXT:NEXT
2160 ZZ=INT(RND(0)*(H))
2170 ZX=INT(MX(ZZ)/10):ZY=MX(ZZ)-ZX*10
2180 RETURN
2200 FORY=0TO3:FORX=0TO3:BW=0
2210 Z=BE(X,Y)+XX:AD=1575+2*X-80*Y:POKEA
D,160:POKEAD+54272,CO(PL)
2220 IFZ>3THENBW=-10000:GOTO2350
2230 BW=BW+RX(Y,Z)+RY(X,Z)+RZ(X,Y)
2240 IFY=ZTHENBW=BW+UX(X)
2250 IFX=ZTHENBW=BW+UY(Y)
2260 IFX=YTHENBW=BW+UZ(Z)
2270 IFY=3-ZTHENBW=BW+OX(X)
2280 IFX=3-ZTHENBW=BW+OY(Y)
2290 IFX=3-YTHENBW=BW+OZ(Z)
2300 IF(X=Y)AND(X=Z)THENBW=BW+D1
2310 IF(X=3-Y)AND(X=Z)THENBW=BW+D2
2320 IF(X=Y)AND(X=3-Z)THENBW=BW+D3
2330 IF(Y=Z)AND(X=3-Z)THENBW=BW+D4
2340 H0=BW*1E4-INT(BW*1E4)
2345 IFBW>64THENBW=64
2346 IFABS(H0-PL/10)<0.05THENBW=65
2350 BW(X,Y)=BW
2360 POKEAD,32
2370 NEXT:NEXT
2380 RETURN
2500 Z=BE(X,Y):BE(X,Y)=Z+1
2505 F=1114:D=3:GOSUB3800
2510 FE(X,Y,Z)=PL
2520 IFZ<>3THENFE(X,Y,Z+1)=1
2530 PRINTCO$(PL):X$=X$(Z):GOSUB3000
2540 H=1:Q=0:FORI=0TO3
2541 H9=FE(I,Y,Z):X(I)=I:Y(I)=Y:Z(I)=Z
2542 GOSUB2800:NEXT
2543 GOSUB2900:RX(Y.Z)=H
2550 H=1:Q=0:FORI=0TO3
2551 H9=FE(X,I,Z):X(I)=X:Y(I)=I:Z(I)=Z
2552 GOSUB2800:NEXT
2553 GOSUB2900:RY(X,Z)=H
2560 H=1:Q=0:FORI=0TO3
2561 H9=FE(X,Y,I):X(I)=X:Y(I)=Y:Z(I)=I
2562 GOSUB2800:NEXT
2563 GOSUB2900:RZ(X,Y)=H
2570 IFY<>ZTHEN2580
```

Listing »3D-Vier gewinnt« (Fortsetzung)

```
2571 H=1:Q=0:FORI=0TO3
2572 H9=FE(X,I,I):X(I)=X:Y(I)=I:Z(I)=I
2573 GOSUB2800:NEXT
2574 GOSUB2900:UX(X)=H
2580 IFX<>ZTHEN2590
2581 H=1:Q=0:FORI=0TO3
2582 H9=FE(I,Y,I):X(I)=I:Y(I)=Y:Z(I)=I
2583 GOSUB2800:NEXT
2584 GOSUB2900:UY(Y)=H
2590 IFX<>YTHEN2600
2591 H=1:Q=0:FORI=0TO3
2592 H9=FE(I,I,Z):X(I)=I:Y(I)=I:Z(I)=Z
2593 GOSUB2800:NEXT
2594 GOSUB2900:UZ(Z)=H
2600 IFY<>3-ZTHEN2610
2601 H=1:Q=0:FORI=0TO3
2602 H9=FE(X,I,3-I):X(I)=X:Y(I)=I:Z(I)=3
-I
2603 GOSUB2800:NEXT
2604 GOSUB2900:OX(X)=H
2610 IFX<>3-ZTHEN2620
2611 H=1:Q=0:FORI=0TO3
2612 H9=FE(I,Y,3-I):X(I)=I:Y(I)=Y:Z(I)=3
-I
2613 GOSUB2800:NEXT
2614 GOSUB2900:OY(Y)=H
2620 IFX<>3-YTHEN2630
2621 H=1:Q=0:FORI=0TO3
2622 H9=FE(I,3-I,Z):X(I)=I:Y(I)=3-I:Z(I)
=Z
2623 GOSUB2800:NEXT
2624 GOSUB2900:OZ(Z)=H
2630 IF(X<>Y)OR(X<>Z)THEN2640
2631 H=1:Q=0:FORI=0TO3
2632 H9=FE(I,I,I):X(I)=I:Y(I)=I:Z(I)=I
2633 GOSUB2800:NEXT
2634 GOSUB2900:D1=H
2640 IF(X<>3-Y)OR(X<>Z)THEN2650
2641 H=1:Q=0:FORI=0TO3
2642 H9=FE(I,3-I,I):X(I)=I:Y(I)=3-I:Z(I)
=I
2643 GOSUB2800:NEXT
2644 GOSUB2900:D2=H
2650 IF(X<>Y)OR(X<>3-Z)THEN2660
2651 H=1:Q=0:FORI=0TO3
2652 H9=FE(I,I,3-I):X(I)=I:Y(I)=I:Z(I)=3
-I
2653 GOSUB2800:NEXT
2654 GOSUB2900:D3=H
2660 IF(Y<>Z)OR(X<>3-Z)THEN2670
2661 H=1:Q=0:FORI=0TO3
2662 H9=FE(3-I,I,I):X(I)=3-I:Y(I)=I:Z(I)
=I
2663 GOSUB2800:NEXT
2664 GOSUB2900:D4=H
2670 Z=Z+1:IFZ=4THEN2700
2680 RX(Y,Z)=RX(Y,Z)*2
2681 RY(X,Z)=RY(X,Z)*2
2682 IFY=ZTHENUX(X)=UX(X)*2
2683 IFX=ZTHENUY(Y)=UY(Y)*2
2684 IFX=YTHENUZ(Z)=UZ(Z)*2
2685 IFY=3-ZTHENOX(X)=OX(X)*2
2686 IFX=3-ZTHENOY(Y)=OY(Y)*2
2687 IFX=3-YTHENOZ(Z)=OZ(Z)*2
2688 IF(X=Y)AND(X=Z)THEND1=2*D1
2689 IF(X=3-Y)AND(X=Z)THEND2=2*D2
2690 IF(X=Y)AND(X=3-Z)THEND3=2*D3
2691 IF(Y=Z)AND(X=3-Z)THEND4=2*D4
2700 RETURN
2800 IFH9=0THENH=H/2:RETURN
2810 IFH9=1THENRETURN
2820 IF(Q<2)OR(H9=Q)THENH=H*4:Q=H9:RETUR
N
2830 H=0:RETURN
2900 IFH=64THENH=H+PL/1E5
2910 IFH<>256THENRETURN
2920 FORJ=0TO3:GX(J)=X(J):GY(J)=Y(J):GZ(
J)=Z(J):NEXT
2930 GW=PL:RETURN
3000 X0=6*X+Z+1:Y0=20-5*Y-Z
3010 POKE211,X0:POKE214,Y0:SYS58732
3020 PRINTX$;:RETURN
3100 POKE214,6:POKE211,30:SYS58732
3110 PRINT"┌──┬──┬──┬──┐"
3120 GOSUB3180:PRINT"│  │  │  │  │"
3130 FORI=1TO3
3140 GOSUB3180:PRINT"├──┼──┼──┼──┤"
3150 GOSUB3180:PRINT"│  │  │  │  │":NEXT
3160 GOSUB3180:PRINT"└──┴──┴──┴──┘"
3170 RETURN
3180 POKE211,30:SYS58732:RETURN
3500 X$=CHR$(144)+CHR$(18)
3510 IFGW<>0THEN3540
3520 X$=X$+LEFT$(BL$,13)+"UNENTSCHIEDEN!
"+LEFT$(BL$,13)
3530 GOTO3580
3540 H$="SIEGER:  "+SP$(GW):H=LEN(H$)
3550 IFH>40THENH$=LEFT$(H$,40):H=40
3560 H=(40-H)/2
3570 X$=X$+LEFT$(BL$,H)+H$+LEFT$(BL$,H+.
5)
3580 PRINTCHR$(19);X$:PRINTCHR$(19)
3590 F=4455:D=7:GOSUB3800
3591 F=5001:D=7:GOSUB3800
3592 F=5613:D=7:GOSUB3800
3593 F=5947:D=20:GOSUB3800
3594 F=4455:D=20:GOSUB3800
3595 F=5947:D=20:GOSUB3800
3596 F=4455:D=20:GOSUB3800
3597 F=5947:D=50:GOSUB3800
3600 IFGW<>0THEN3620
3610 POKE198,0:WAIT198,1:POKE198,0:GOTO1
00
3620 PRINTCO$(GW):GOSUB3750
3630 T=TI
3640 GETK$:IFK$<>""THEN100
3650 IFTI-T<30THEN3640
3660 PRINTCL$(GW):GOSUB3750
3670 T=TI
3680 GETK$:IFK$<>""THEN100
3690 IFTI-T<30THEN3680
3700 GOTO3620
3750 FORI=0TO3
3760 X=GX(I):Y=GY(I):Z=GZ(I):H=0
3770 IFZ=0THENH=H+1
3780 IFZ+1=BE(X,Y)THENH=H+2
3790 X$=W$(H):GOSUB3000
3795 NEXT:RETURN
3800 F0=INT(F/256):POKE54272,F-256*F0
3810 POKE54273,F0:POKE54276,65
3820 T=TI
3830 IFTI-T<DTHEN3830
3840 POKE54276,0:RETURN
READY.
```

Listing »3D-Vier gewinnt« (Schluß)



Druckfehlerteufelchen



Hardcopy mit dem VC 1520, Ausgabe 7/84, Seite 108

Die korrekte Zeile lautet: POKE44,64:POKE64*256, 0:NEW
und nicht 64-256!

Einzeiler, Ausgabe 11/84, Seite 155

Die Spaces bei dem Einzeiler DI-AS müssen als Shift-Space eingegeben werden.

Die Ebenen des Absturzes, 11/84, Seite 93

Das letzte Datum in Zeile 360 (Programm UNNEW) lautet 255.

User-Port-Tastatur, Ausgabe 10/84, Seite 93

Das Datum 28 in Zeile 880 muß in 82 umgewandelt werden.

FLIST, Ausgabe 10/84, Seite 90

Der Hinweis für Druckerbesitzer muß natürlich lauten:
OPEN1,4:CMD1:FLIST
und nach dem Drucken: PRINT#1:CLOSE1

Außerdem bin ich durch Zufall auf einen Programmfehler gestoßen, der bei aktivem Programm auftritt und bei einer Variablenzuweisung, bei der die Variable mit F beginnt einen SYNTAX ERROR erzeugt (zum Beispiel bei F=1, FX=100 etc.).

Dies kann man auf zwei Arten verhindern:

Entweder schaltet man die Erweiterung vor dem Starten eines Basic-Programms durch SYS 58451 ab, oder man ändert den Basic-Lader wie folgt:

Den Endwert der FOR-NEXT-Schleife auf 981 setzen, in Zeile 230 den letzten Wert 8 durch 195 und in Zeile 240 den ersten Wert 175 in 3 abändern.

Dann fügt man diese DATA-Zeilen an:

```
440 DATA 56,165,122,233
450 DATA 1,133,122,165
460 DATA 123,233,0,133
470 DATA 123,32,121,0
480 DATA 76,231,167
```

Michael Weidlich

So macht man Basic-Programme schneller, Ausgabe 10/84, Seite 55

Bei den Versionen 1 für C 64 und VC 20 wurde in den Zeilen 40 jeweils nach dem Z ein »,1« vergessen. Also: 40 POKE 7680 + Z,1: POKE 38400+Z,6 etc.
Seite 58, Version 13: 40 PRINT"A";

MSD-Super-Disk-Drive, Ausgabe 11/84, Seite 15

Die vollständige Bezugsadresse von Softline lautet: Softline, Schwarzwaldstraße 8a, 7602 Oberkirch, Tel.: 07802/3707

Titelfoto, 11/84, Seite 1

Das Titelfoto dieser Ausgabe wurde von Limelight Studio, Karolinenstraße 3, 8000 München 22, Tel.: 089/222397 erstellt.

Wollen Sie einen gebrauchten Computer verkaufen oder erwerben? Suchen Sie Zubehör? Haben Sie Software anzubieten oder suchen Sie Programme oder Verbindungen? Die FUNDGRUBE von 64'er bietet allen Computerfans die Gelegenheit, für nur DM 5,— eine private Kleinanzeige mit bis zu 5 Zeilen Text in der Rubrik Ihrer Wahl aufzugeben. Und so kommt Ihre private Kleinanzeige in die FUNDGRUBE der **Januar-Ausgabe** (erscheint am 14. Dez. 84): Schicken Sie Ihren Anzeigentext bis zum 20. Nov. 84 (Datum des Poststempels und Anzeigenschluß) an »64'er«. Später eingehende Aufträge werden in der **Februar-Ausgabe** (erscheint am 20. Dez. 84) veröffentlicht.

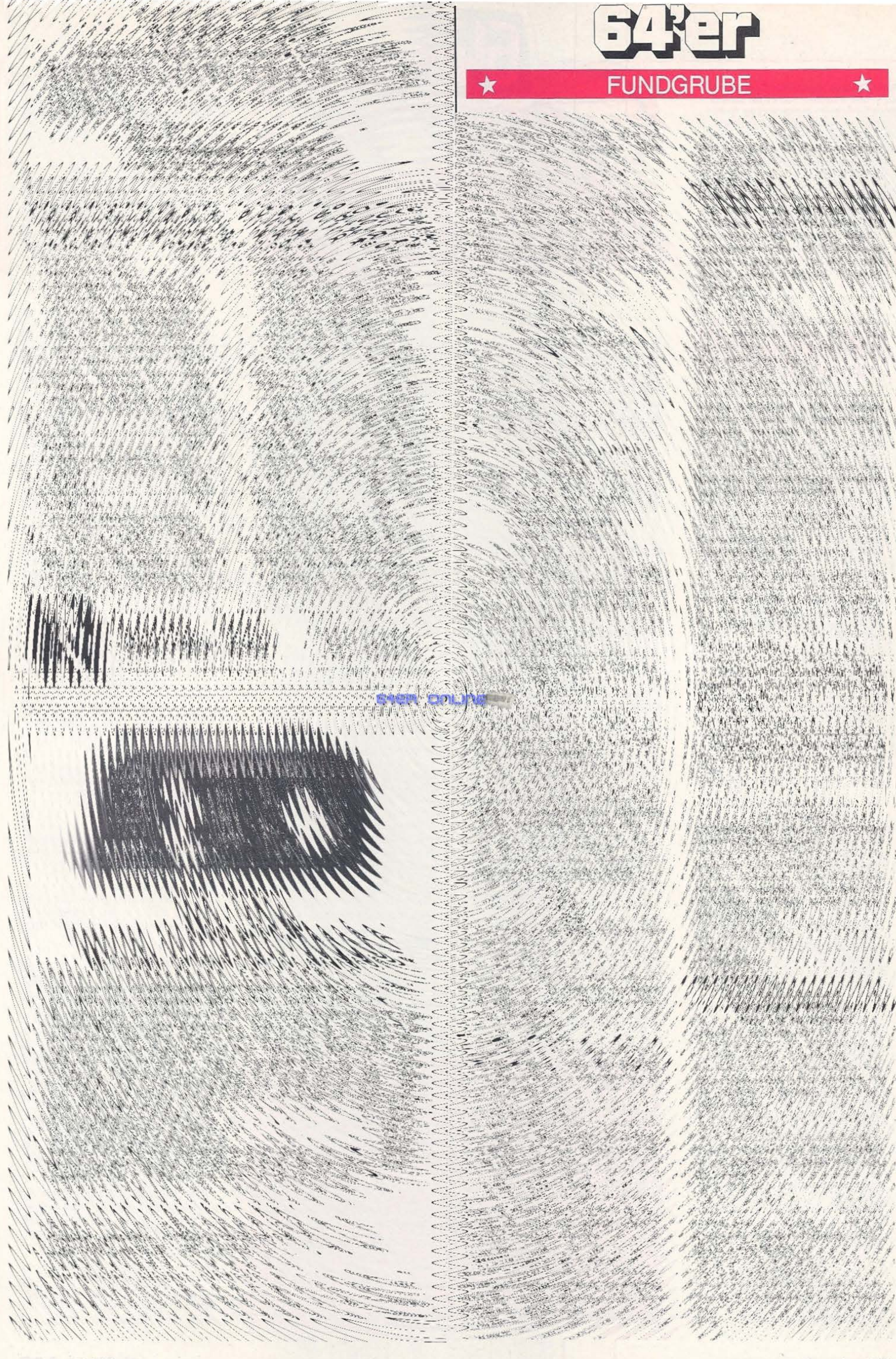


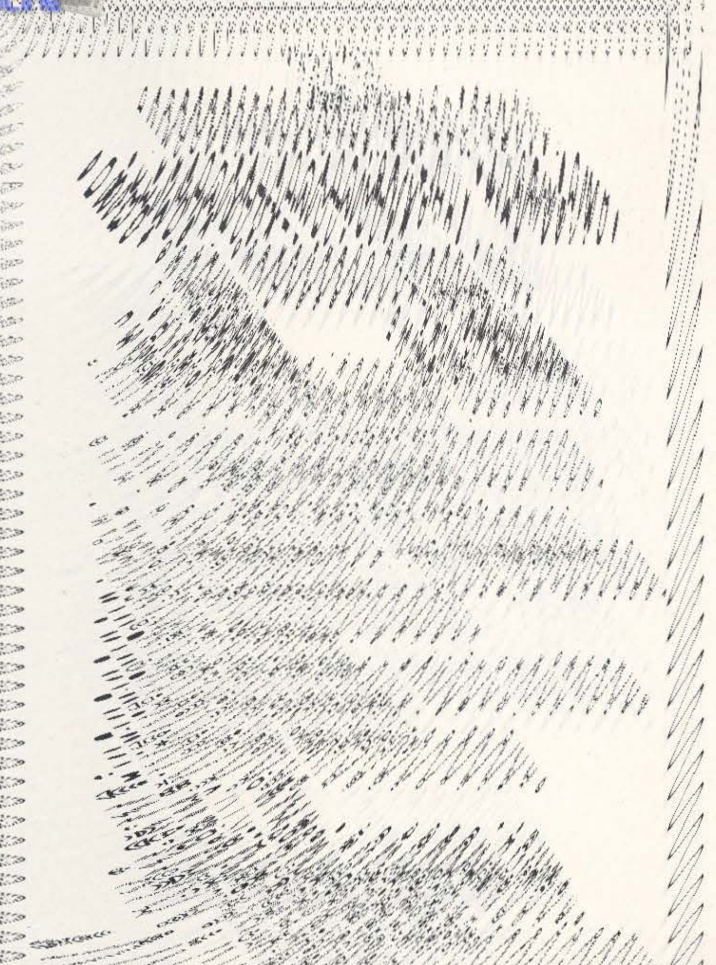
Am besten verwenden Sie dazu die vorbereitete Auftragskarte am Anfang des Heftes. Bitte beachten Sie: Ihr Anzeigentext darf maximal 5 Zeilen mit je 32 Buchstaben betragen. Überweisen Sie den Anzeigenpreis von DM 5,— auf das Postscheckkonto Nr. 14 199-803 beim Postscheckamt mit dem Vermerk »Markt & Technik, 64'er«, oder schicken Sie uns DM 5,— als Scheck oder in Bargeld.

Der Verlag behält sich die Veröffentlichung längerer Texte vor. Kleinanzeigen, die entsprechend gekennzeichnet sind oder deren Text auf eine gewerbliche Tätigkeit schließen läßt, werden in der Rubrik »Gewerbliche Kleinanzeigen« zum Preis von DM 10,— je Zeile Text veröffentlicht.

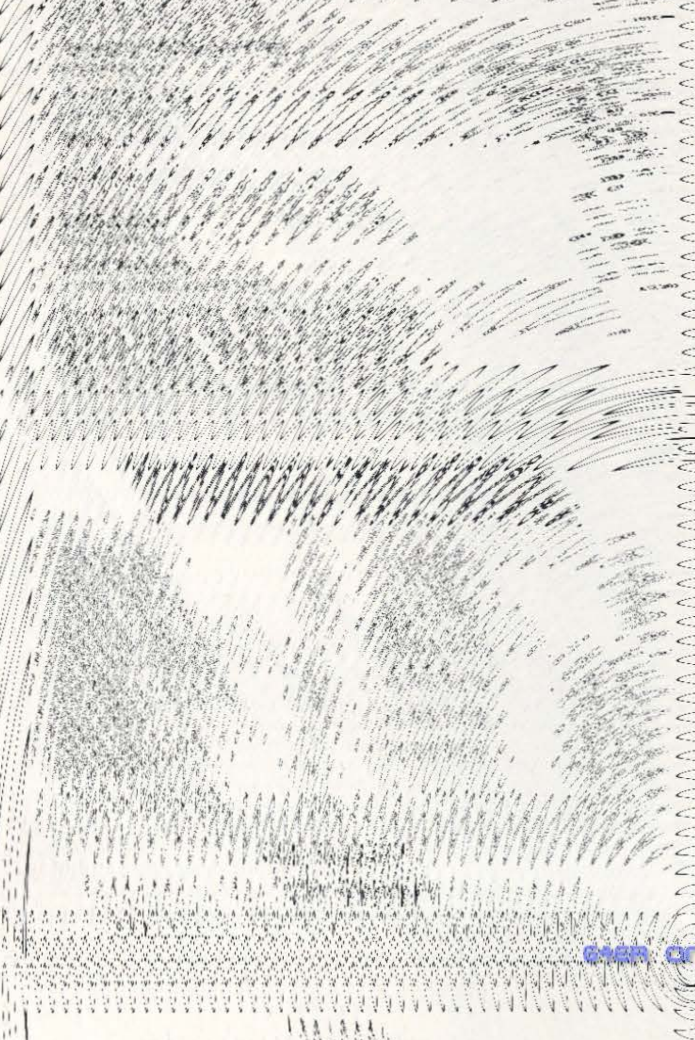


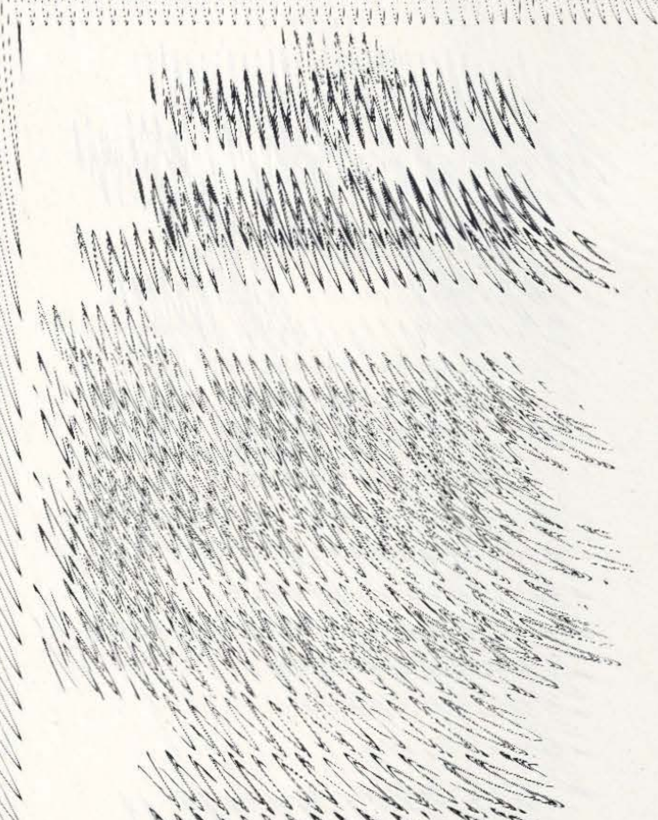




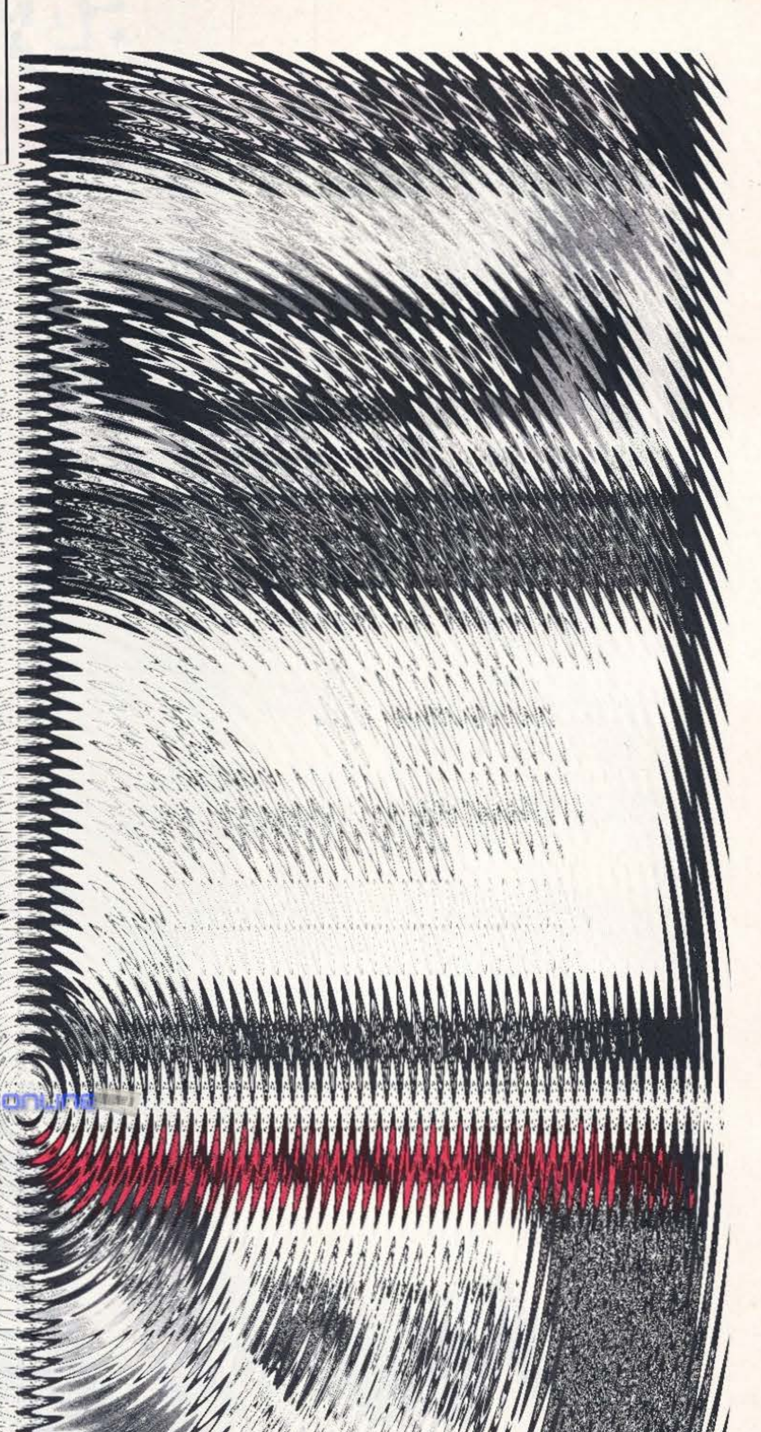




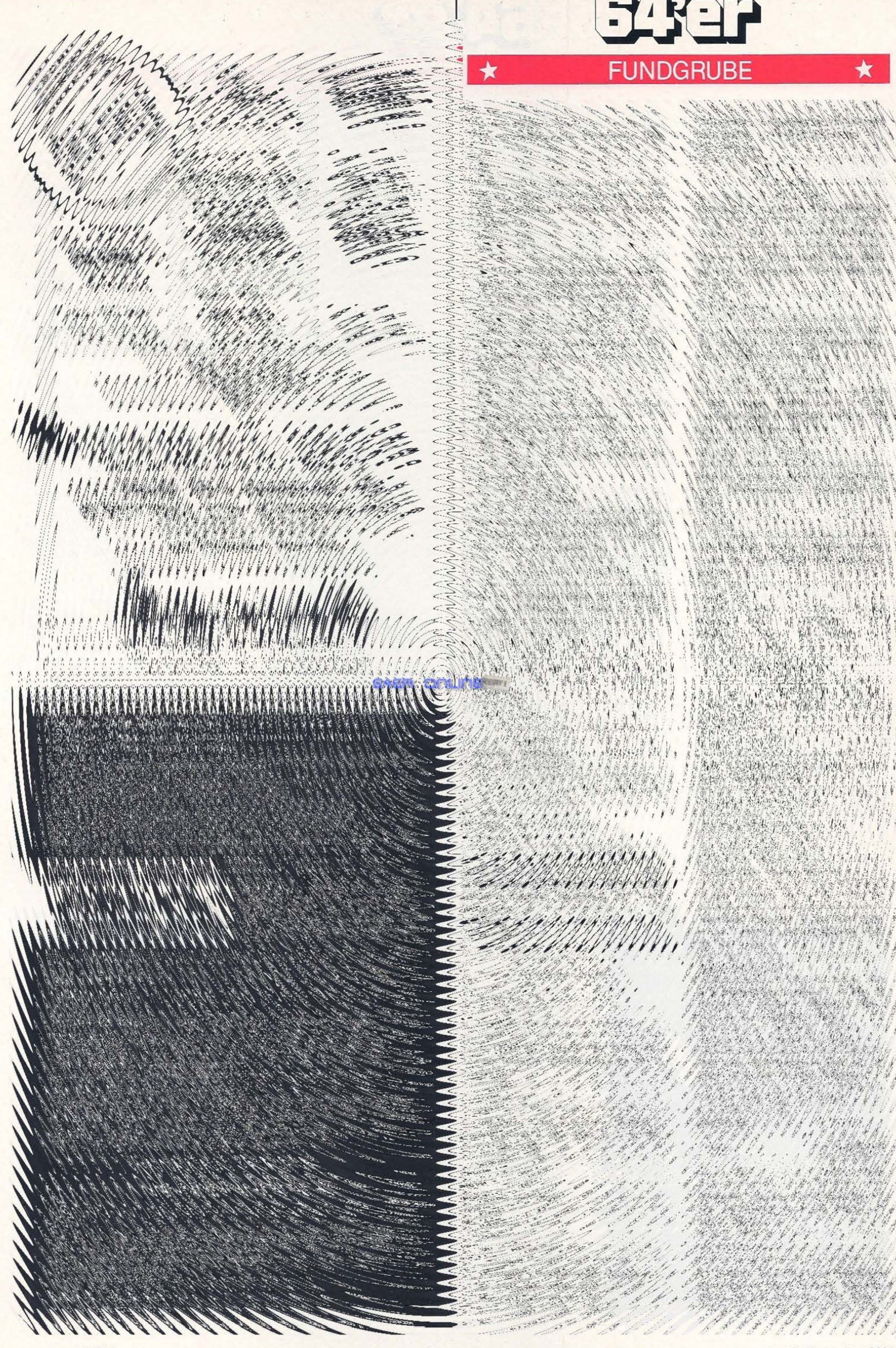


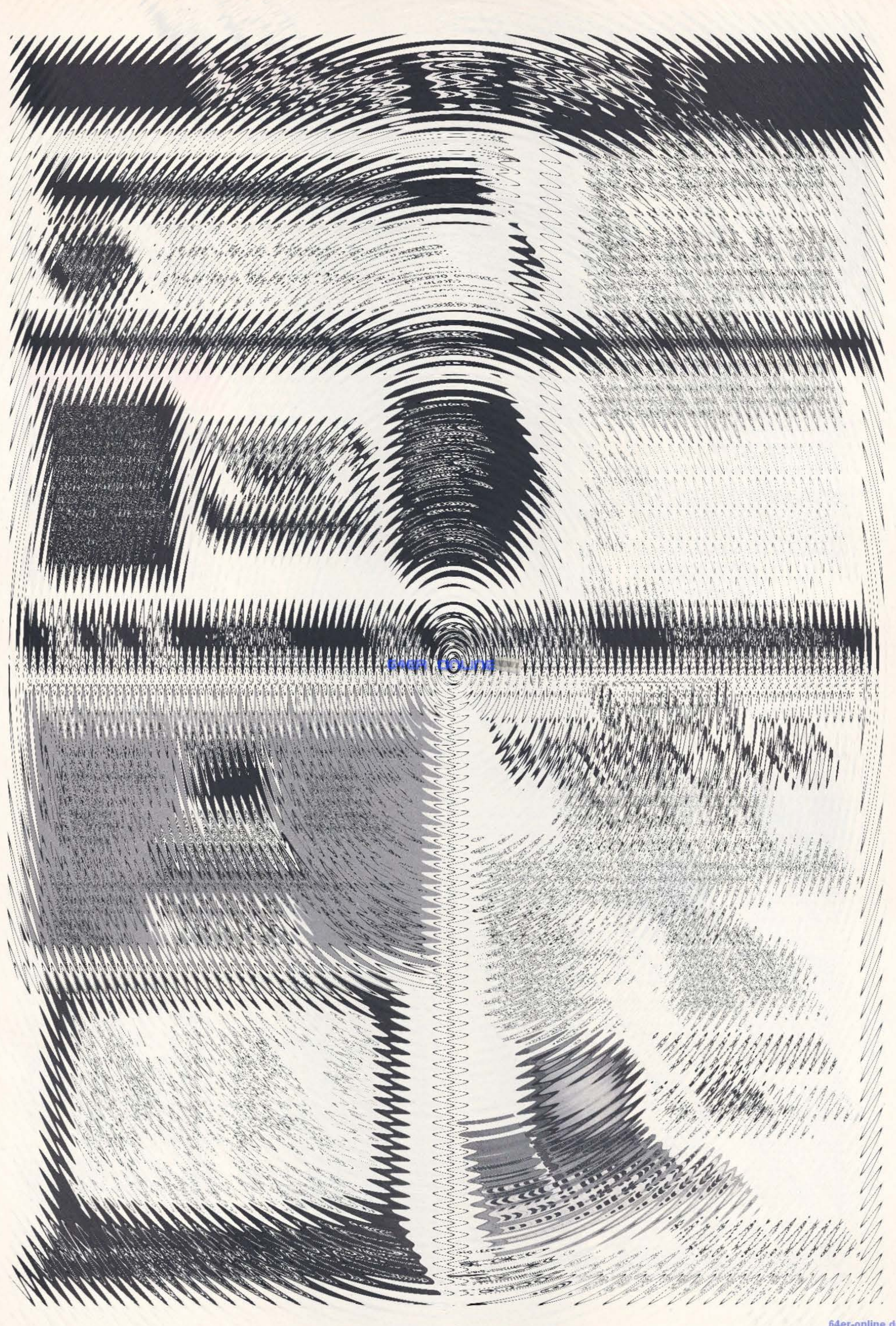


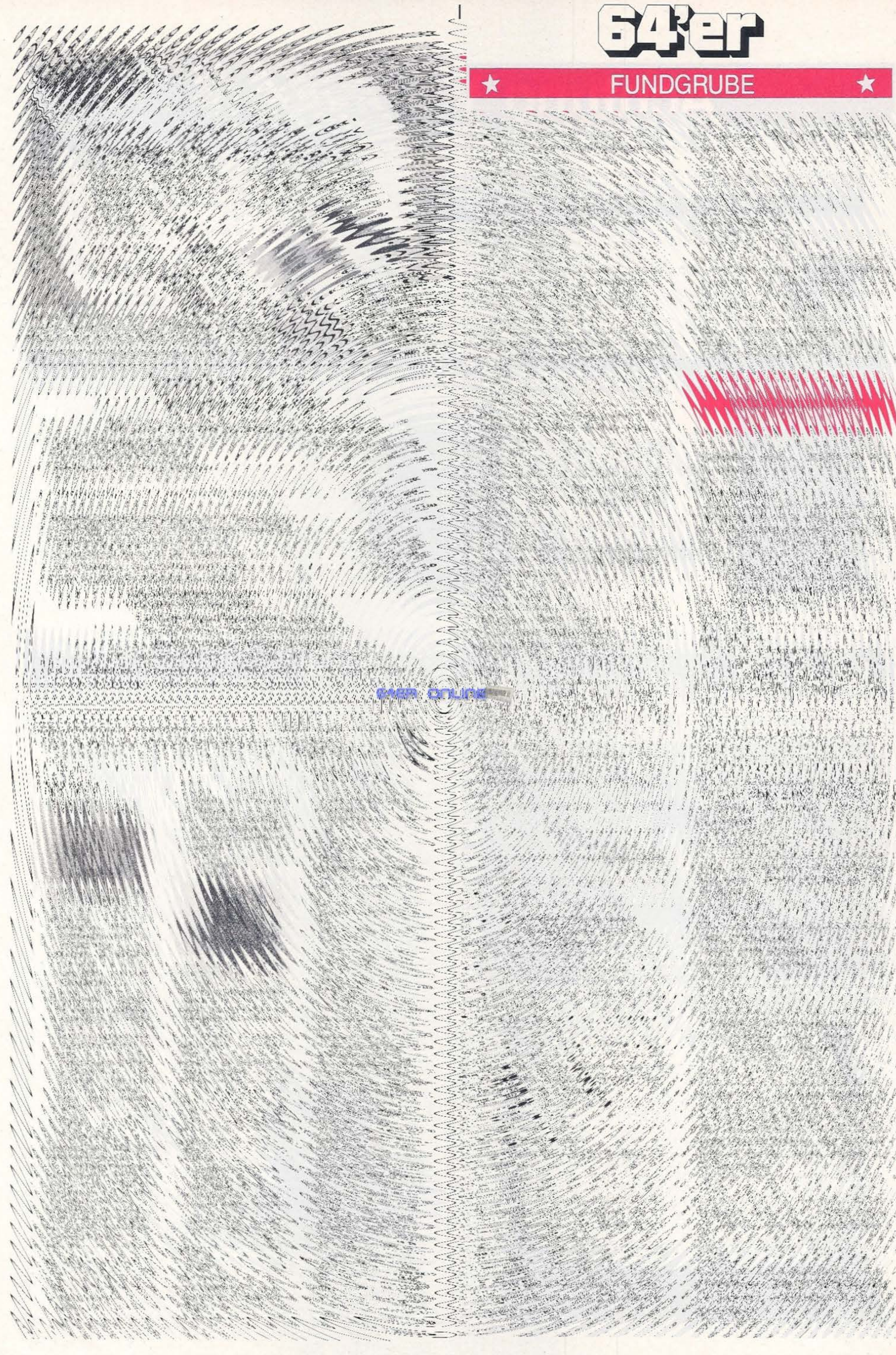


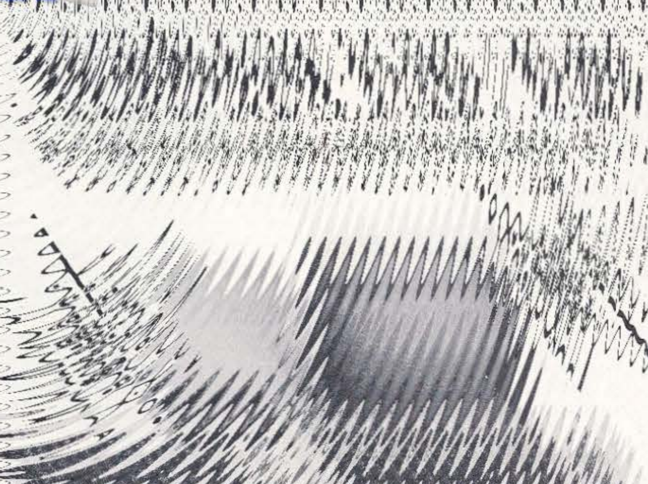


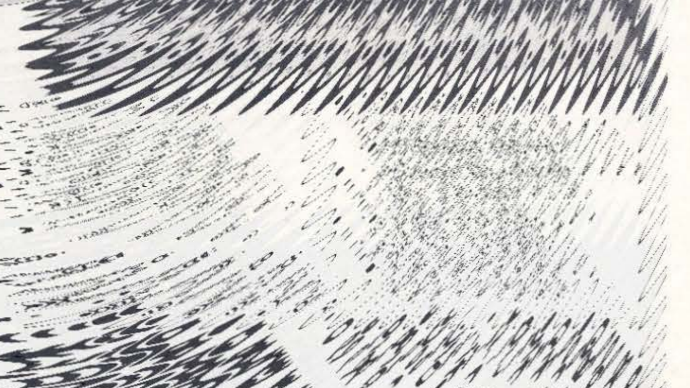


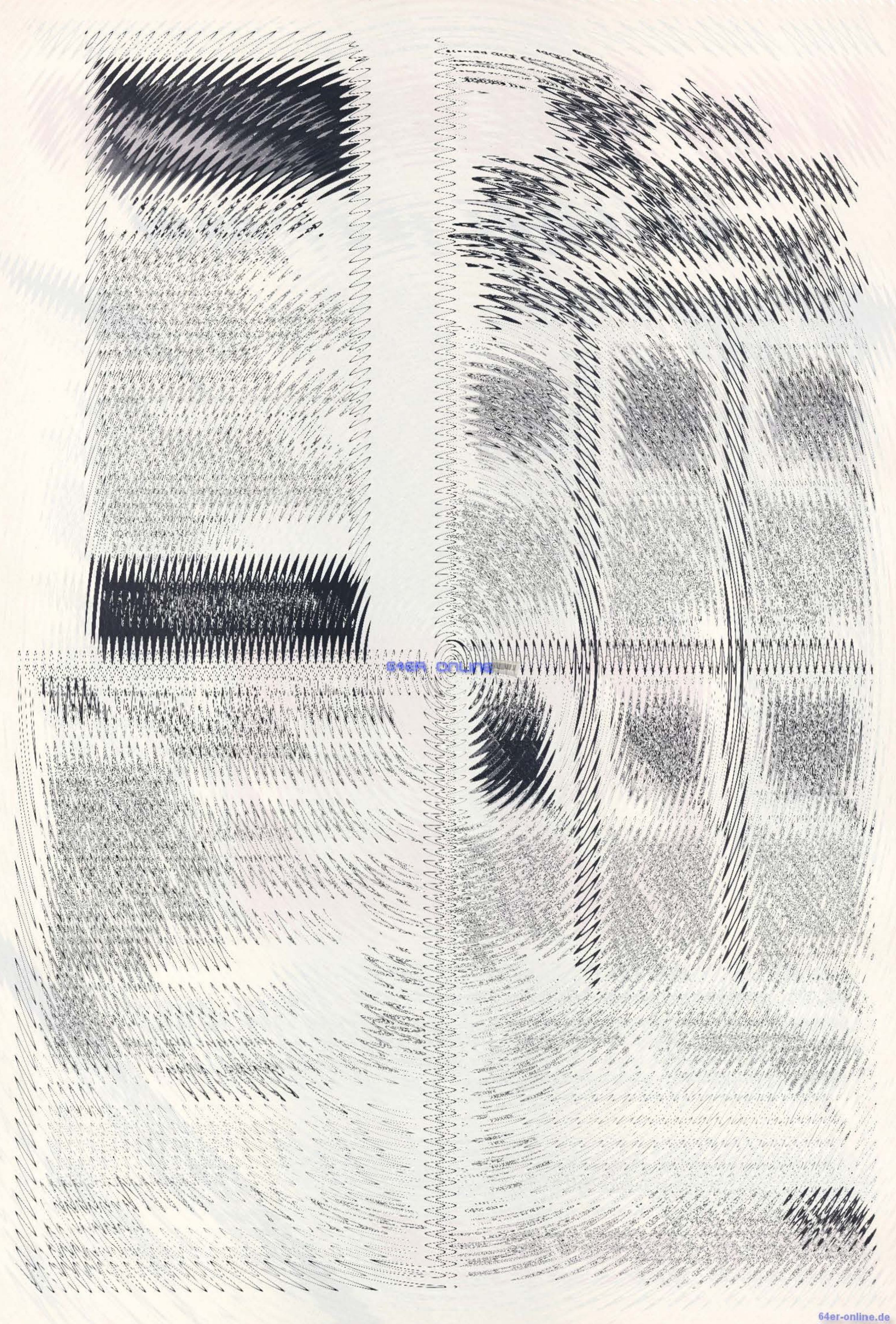


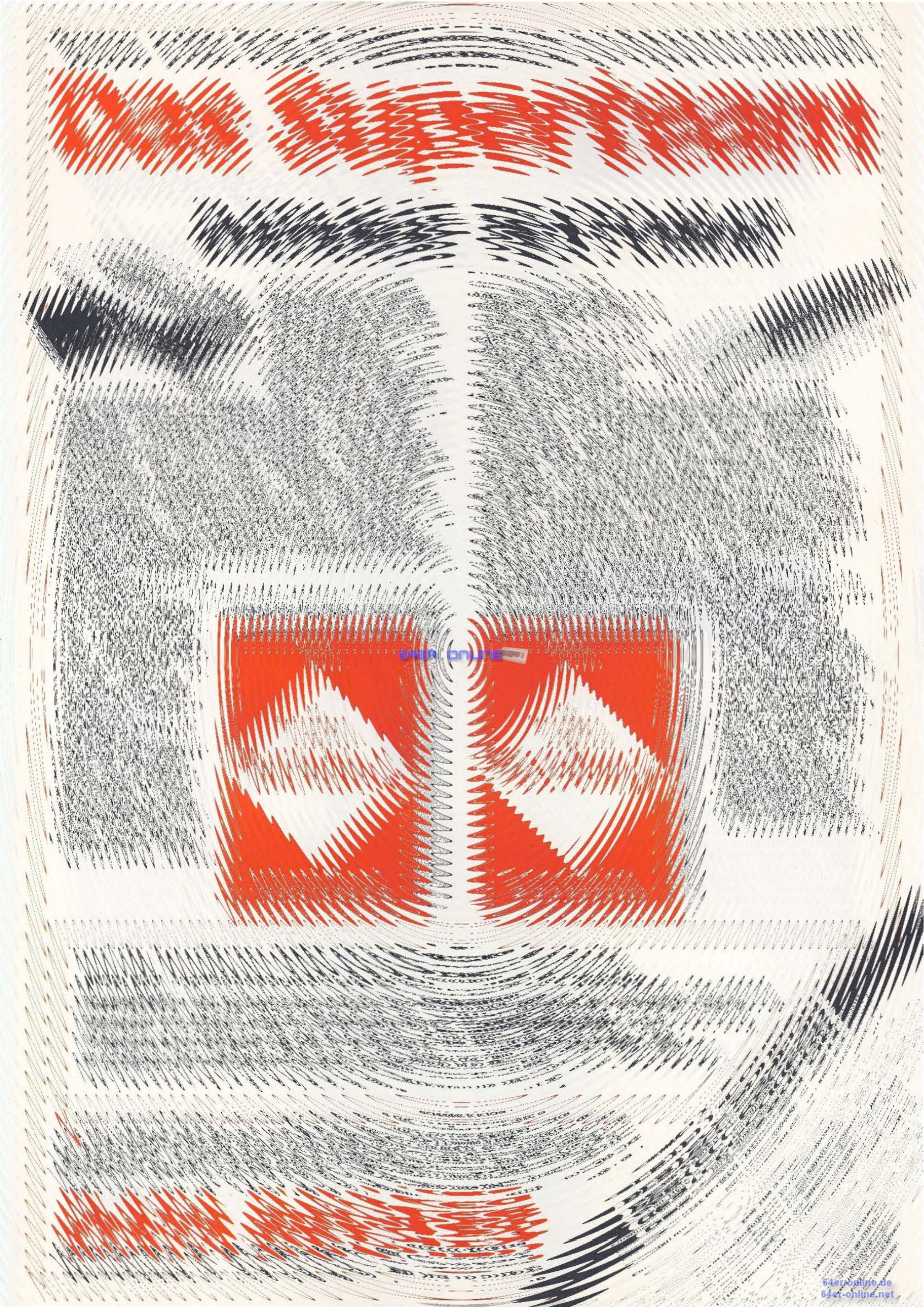




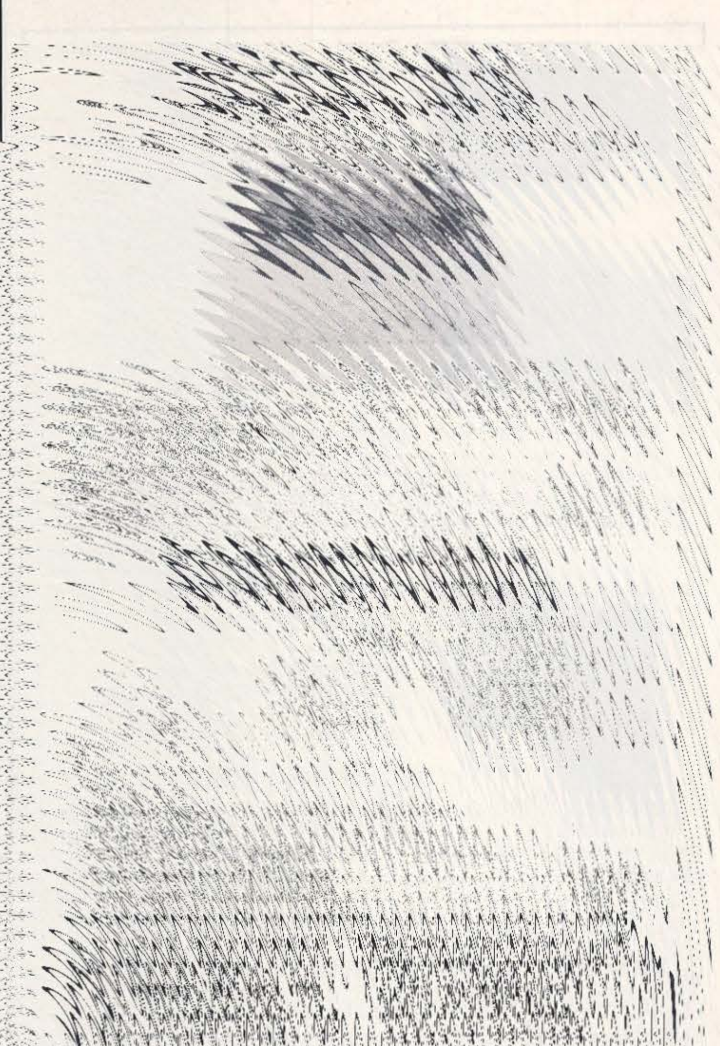


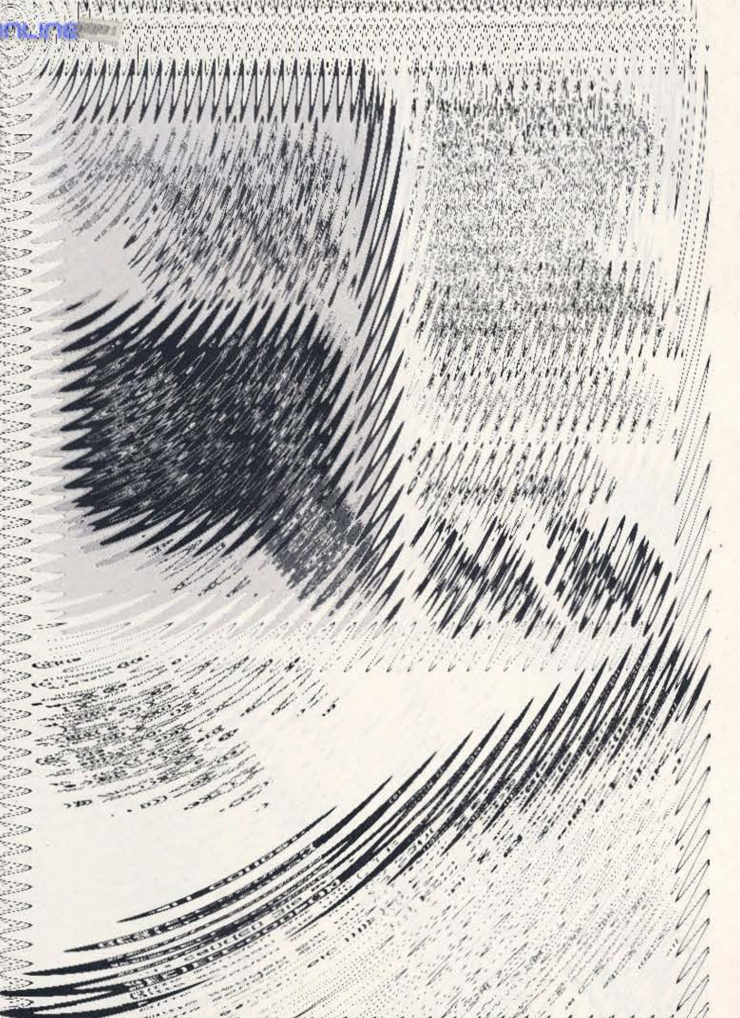


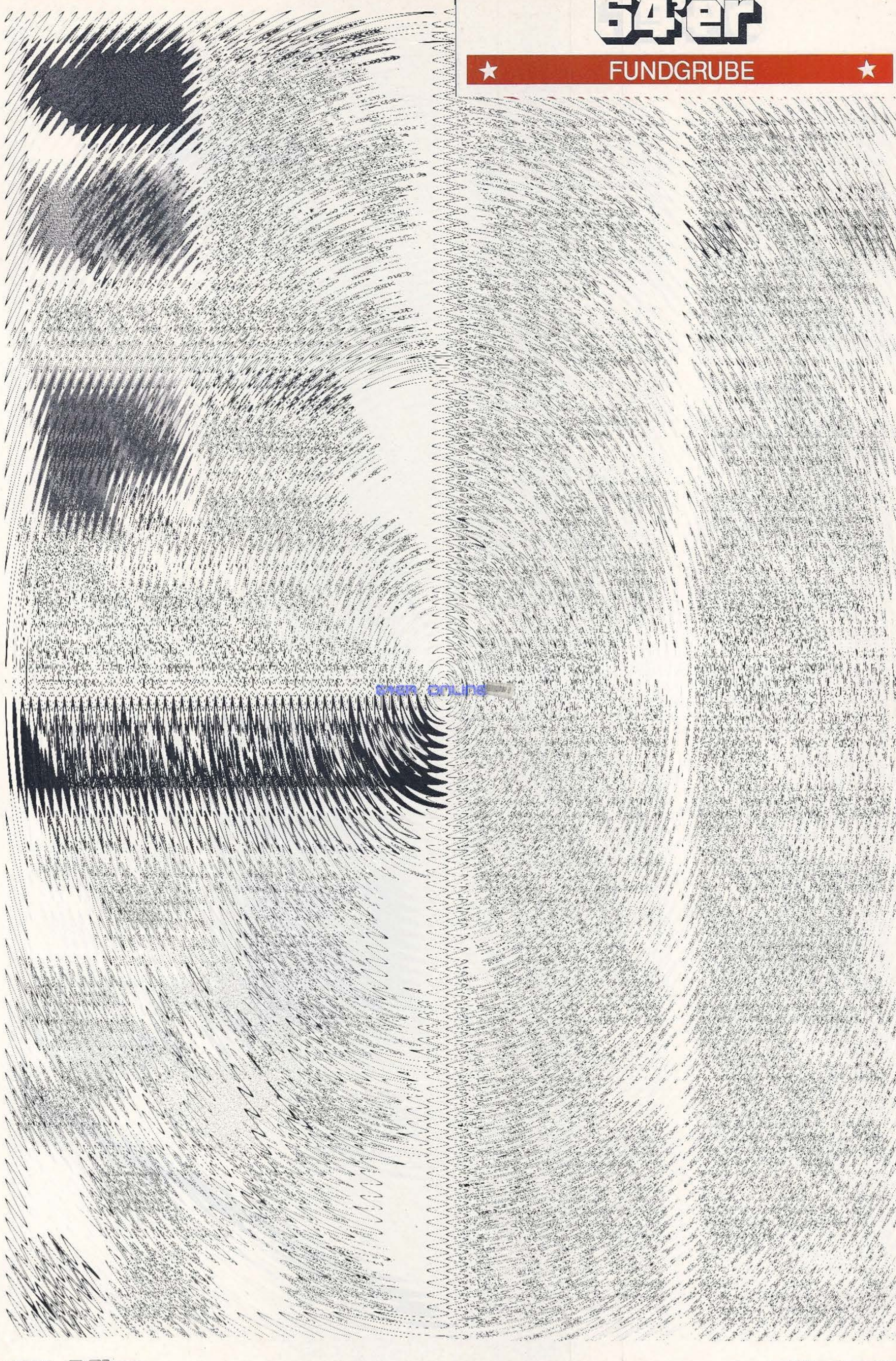




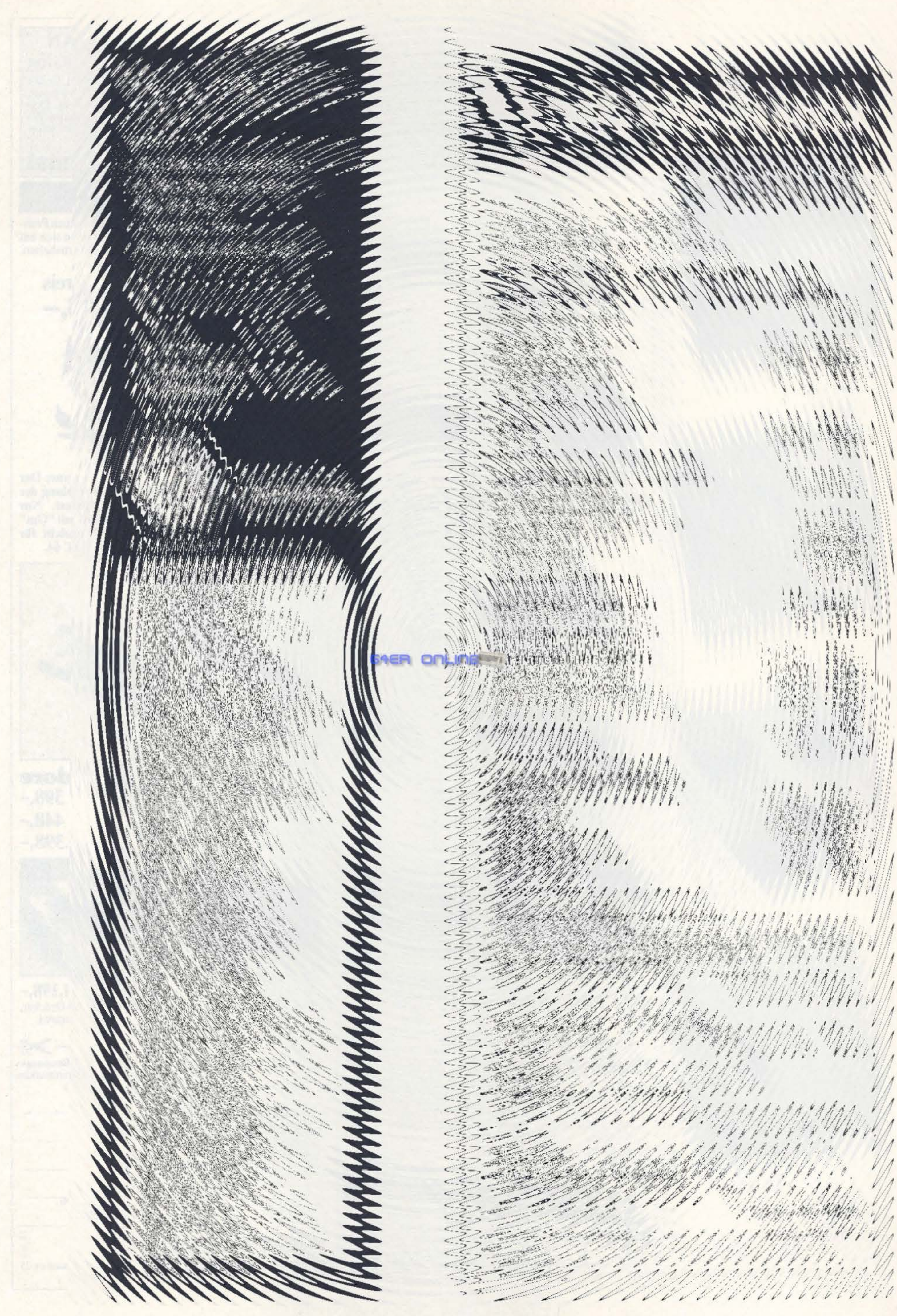








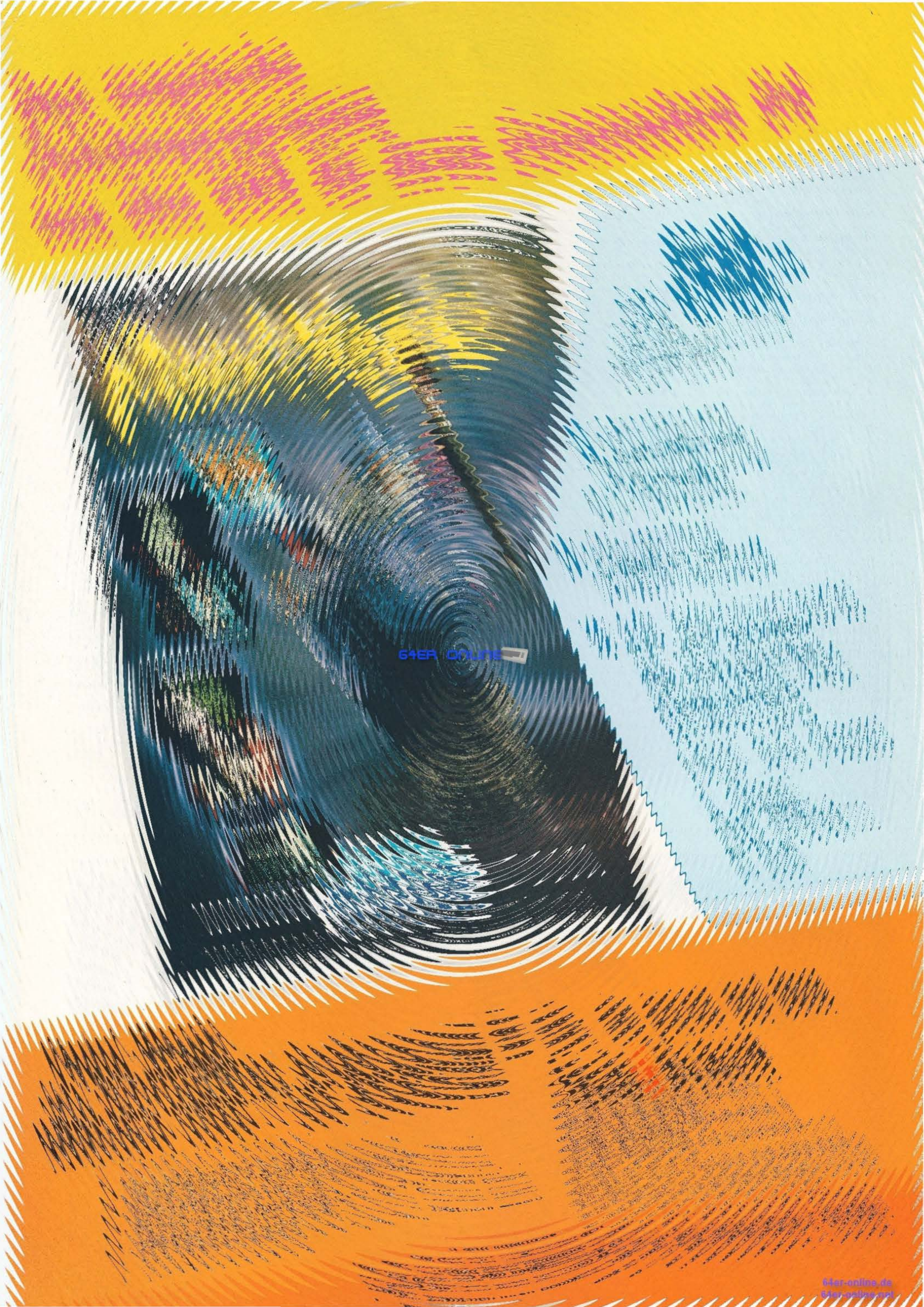












DEM KLANG AUF DER SPUR

Teil I

Im C 64 steckt jede Menge Musik. Doch wie man dem Heimcomputer die richtigen Klänge entlockt, wissen viele nicht. Sie erfahren es in diesem Kurs.

Zunächst wollen wir ein wenig auf die Hardware und auf die Grundzüge der Tonerzeugung eingehen. Zuständig für den Sound des C 64 ist ein unscheinbares kleines Chip, das als SID (Sound Interface Device) bezeichnet wird. Dieser hochentwickelte Synthesizerbaustein enthält drei Stimmen, die ein Synthi-Freak als DCOs (Digital Controlled Oscillators) bezeichnet. Sie realisieren polyphone (mehrstimmige) Klänge oder Sequenzen. Für jeden dieser Oszillatoren stehen vier Wellenformen zur Verfügung (Bild 1).

Wellenformen

Ein gespielter Ton besteht aus einer Grundfrequenz, sowie den Obertönen. Die Tonhöhe wird durch die Grundfrequenz bestimmt. Oberwellen sind Sinusschwingungen, deren Frequenzen ein ganzzahliges Vielfaches der Grundfrequenz sind. Ein monophoner Klang besteht also aus einer Grundfrequenz und seinen Obertönen. Eine Ausnahme hiervon bildet das Rauschen, bei dem es sich um ein Frequenzgemisch handelt.

Ein akustisches Instrument, wie etwa ein Klavier oder eine Trompete, hat eine komplizierte Oberwellenstruktur. Diese kann sich beim Spielen auch ändern, zum Beispiel bei einem Klavier: Der Anschlag eines Klaviertones ist relativ obertonreich, die Note klingt dann jedoch dumpf aus. Im folgenden möchten wir auf die im SID vorhandenen Wellenformen kurz eingehen:

1. Die Dreieckschwingung:
Diese Wellenform ist der Sinusschwingung sehr ähnlich. Die Dreieckswelle enthält ebenfalls sehr wenig Obertöne. Der Klang erinnert an eine Holzflöte.

2. Die Sägezahnschwingung:
Diese Welle enthält alle Oberwellen und klingt daher sehr hell. Sie erinnert vom Klang her an eine Violine.

3. Die Rechteckschwingung:
Dies ist eigentlich die vielseitigste Wellenform. Durch Änderung der Pulsbreite lassen sich extreme Klangveränderungen erzielen. Der Grundton erinnert an eine Klarinette.

4. Weißes Rauschen:
Dieses Frequenzgemisch läßt sich insbesondere in Verbindung mit dem Filter sehr vielseitig für Geräuscheffekte einsetzen. Beispielsweise zur Imitation eines Schlagzeugs.

Filter

Zur Beeinflussung eines Klanges fehlt jetzt noch ein Filter. Er verändert den Oberwellengehalt einer Welle. Der SID besitzt drei verschiedene solcher Filter: einen Tiefpaßfilter, einen Bandpaßfilter und einen Hochpaßfilter. Diese werden unabhängig voneinander oder auch kombiniert verwendet.

1. Tiefpaßfilter:
Dieser Filter läßt alle tieferen Töne passieren, während er die hohen Freqenzen abschwächt. Dieser »Effekt« wird durch eine variierbare Grenzfrequenz des Filters geregelt. Man bestimmt mit ihm, wieviele Obertöne zum Beispiel aus einer Sägezahnschwingung herausgefiltert werden. Durch eine Filterung der Obertöne lassen sich Instrumente gut imitieren.

2. Hochpaßfilter:
Dieser Filtertyp schwächt die tieferen Töne stark ab (nämlich wie der Tiefpaß genau um 12 dB). Durch Hochpaßfilter geformte Klänge klingen sehr »dünn«, da ihnen jegliches Baßfundament fehlt.

3. Bandpaßfilter:
Der Bandpaßfilter läßt nur ein schmales Frequenzband hindurch, während er alle übrigen Frequenzen abdämpft. Die Klangergebnisse sind ähnlich wie beim Hochpaß, also sehr dünn. Dieser Filter besitzt eine Flankensteilheit von nur 6 dB.

4. Notchfilter:
Der Notchfilter ist eine Kombination aus Hochpaß- und Tiefpaßfilter und gilt deshalb meist nicht als eigenständiger Filtertyp. Er heißt auch Bandsperre und übernimmt genau die umgekehrte Funktion eines Bandpaßfilters. Er schwächt ein kleines Frequenzband ab, während er alle restlichen Frequenzen durchläßt.

Die Hüllkurve:
Um zum Beispiel ein Klavier zu imitieren, genügt es nicht, den Klang durch Einsatz von Filtern nachzuahmen. Auch der Lautstärkeverlauf eines Instruments muß berücksichtigt werden. Ein charakteristischer Klavierton hat einen harten Anschlag und klingt dann bald aus. Um diesen Lautstärkeverlauf nachzuahmen, bedient man sich in den meisten Synthesizern eines Hüllkurvengenerators (ADSR).

ADSR

ADSR ist die Abkürzung für Attack-Decay-Sustain-Release. Man hat den Lautstärkeverlauf einfach in vier Phasen aufgeteilt. Attack nennt man die Zeit, die ein Ton benötigt, um seine Spitzenlautstärke zu erreichen. Die Zeit, in der der Ton wieder auf ein geringeres Lautstärkeniveau abfällt, nennt man Decay. Den Bereich mit dem geringeren Volumenpegel nennt man Sustain. Soll der Ton nachklingen, wie etwa bei einem Klavier mit gedrucktem Dämpferpedal, dann wird dies durch Release erreicht. Zum besseren Verständnis haben wir alle vier Parameter auf einem Schaubild zusammengefaßt (Bild 2). In der nächsten Ausgabe werden wir intensiv auf den Sound und seiner Programmierung am C 64 eingehen.

(Bernhard Carli/ Christian Spitzner/aa)

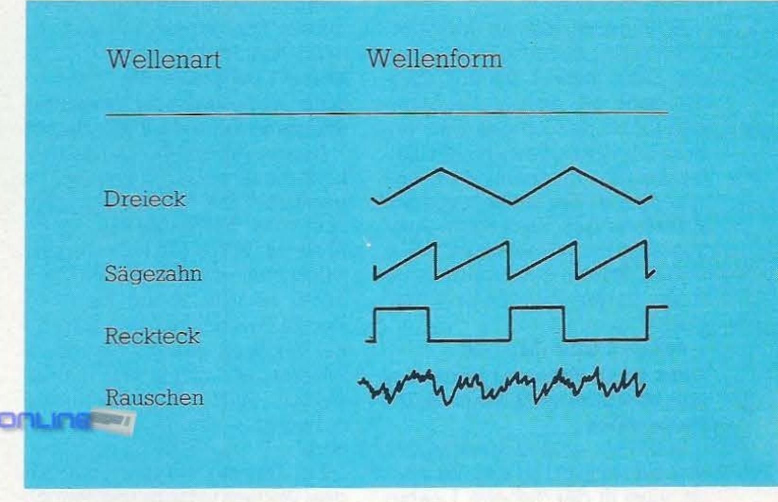


Bild 1. Die verschiedenen Wellenformen

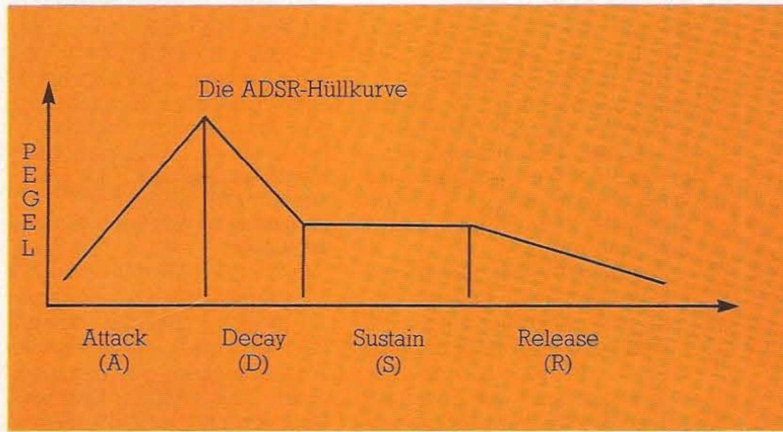


Bild 2. Die ADSR-Hüllkurve

Memory Map mit Wandervorschlägen, Teil 2

Bei der Durchforstung der ersten 1024 Speicherzellen werden wir in dieser Folge die Adressen 3 bis 17 etwas genauer beleuchten.

Das letzte Mal hatten die besprochenen Speicherzellen verschiedene Bedeutungen für VC 20 und C 64.

Ab diesmal, also ab Speicherzelle 3 über mehrere Folgen dieser Serie hinaus bis zur Speicherzelle 672 gelten alle Angaben für beide Computer, zumindest was die Bedeutung der Zellen betrifft. Ihr Inhalt kann entsprechend der verschiedenen Adressen der Betriebssysteme voneinander abweichen. Wie üblich werde ich natürlich jeweils darauf aufmerksam machen.

Adresse 3 und 4 ($3 — $4)

Vektor auf die Routine zur Umwandlung einer Gleitkommazahl in eine ganze Zahl mit Vorzeichen

In diesen beiden Speicherzellen steht also ein Vektor. Was das ist, wird in der Tabelle 1 näher erläutert. Beim VC 20 deutet dieser Vektor auf die Adresse 53674 ($D1AA), beim C 64 auf 45482 ($B1AA). Sie können das mit PRINT PEEK (3)+256*PEEK (4) leicht nachprüfen. Ab diesen Adressen beginnt im Basic-Übersetzer (Interpreter) ein Programm, welches — natürlich in Maschinensprache — eine Gleitkommazahl in eine ganze Zahl umwandelt.

Diejenigen Leser, welche mit Gleitkommazahlen nicht so vertraut sind, möchte ich auf die Tabelle 2 verweisen. Er ist nur eine kleine Einführung. Später, bei der Behandlung der Speicherzellen 97 — 102 werde ich im Detail auf die externe und interne Darstellung und Verwendung von Gleitkommazahlen eingehen.

Dieses Umwandlungsprogramm steht nicht nur den Maschinen, sondern auch den Basic-Programmierern zur Verfügung, allerdings nur über den USR-Befehl und da auch nur, wenn der »Floating Point Accumulator« #1 (FAC1) in den besagten Adressen 97 bis 102 mitbenutzt wird. Ich verschiebe daher alle weiteren Details auf unsere Ankunft bei diesen Speicherzellen.

Bis dahin haben Sie hoffentlich auch den Assemblerkurs weiter verfolgt, die Assembler-, Dissassembler- und Monitorprogramme eingetippt und können damit arbeiten. Dann können wir viel besser den ganzen Zusammenhang verfolgen.

Adresse 5 und 6 ($5 — $6)

Vektor auf die Routine zur Umwandlung einer ganzen Zahl in eine Gleitkommazahl

Dieses Programm ist die Umkehrung der oberen Routine. Es beginnt beim VC 20 ab Speicherzelle 54161 ($D391), beim C 64 ab 45969 ($B391). Da hier prinzipiell dasselbe gilt wie oben, möchte ich nur kurz den Vorteil beleuchten, den derartige Vektoren haben. Eigentlich könnten wir direkt auf die im Vektor enthaltenen Adressen springen — wenn wir sie kennen.

Ein Sprung auf die Adresse des Vektors erlaubt uns jedoch immer die völlige Ignoranz seines Inhalts — und Commodore erlaubt die Änderung der Adressen im Basic-Übersetzer, wie es ja beim C 64 gegenüber dem VC 20 auch gemacht worden ist, ohne daß vorhandene Programme umgeschrieben werden müssen.

Adresse 7 ($7)

Suchzeichen zur Prüfung von Texteingaben in Basic

Diese Speicherzelle wird viel von denjenigen Basic-Routinen als Zwischenspeicher benützt, die den direkt eingegebenen Text absuchen, um Steuerzeichen (Gänsefüße, Kommata, Doppelpunkte und die Zeilenbeendigung durch die RETURN-Taste) rechtzeitig zu erkennen. Normalerweise wird in der Zelle 7 der ASCII-Wert dieser Zeichen abgelegt. Die Speicherzelle 7 wird aber auch von anderen Basic-Routinen benützt. Sie ist daher für den Programmierer praktisch nicht zu verwerten.

Adresse 8 ($8)

Suchzeichen speziell für Befehlsende und Gänsefüße

Wie Speicherzelle 7 dient auch die Zelle 8 als Zwischenspeicher für Basic-Texteingabe und zwar während der Umwandlung von Basic-Befehlen in den vom Computer verwendeten Befehlscode (Tokens). Die Speicherzelle 8 ist vom Programmierer nicht verwertbar.

Adresse 9 ($9)

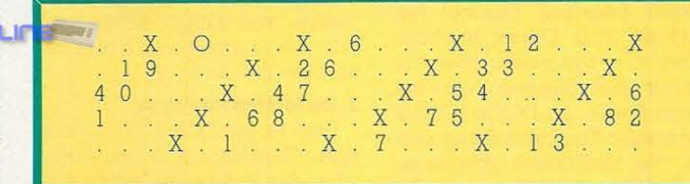
Spaltenposition des Cursors vor dem letzten TAB- oder SPC-Befehl

Speicherzelle 9 wird von den Basic-Befehlen TAB und SPC verwendet. Vor ihrer Ausführung wird die Nummer der Spalte, in der sich der Cursor befindet, aus der Speicherzelle 211 ($D3) nach 9 gebracht, von wo sie geholt wird, um die Position des Cursors nach der Ausführung von TAB und SPC auszurechnen.

Diese komplizierte Erklärung können wir durch Ausprobieren deutlicher machen. Dazu PRINTen wir 16mal den Buchstaben X hintereinander (Semicolon !), allerdings mit SPC (2) jeweils um 2 Spalten versetzt.

```
10 FOR I=0 TO 15
20 PRINT SPC (2) "X";
30 PRINT PEEK (9);
40 NEXT I
```

Nach jedem X wird durch Zeile 30 die »alte« Cursor-Spaltenposition ausgedruckt und zwar in derselben Zeile, ausgelöst durch das Semicolon. Dadurch erhöht sich laufend die in Speicherzelle 9 stehende Positionsangabe des Cursors. Wir erhalten folgenden Ausdruck:



Sie können die Positionsnummer nachrechnen. Berücksichtigen Sie bitte aber dabei, daß bei PRINT vor und nach jeder Zahl eine Stelle frei bleibt, die erste für das Vorzeichen, die zweite wegen des Abstandes.

Wichtig ist außerdem, daß die maximal mögliche Spaltenzahl nicht die Bildschirmspaltenzahl, sondern die »logische« Spaltenzahl ist, also 88 beim VC 20 und 80 beim C 64.

Wir können die Cursorposition in Adresse 9 auch abfragen und ein Programm damit steuern. Fügen Sie einfach in das obige Programm die folgende Zeile 35 ein:

```
35 IF PEEK (9)=33 THEN PRINT
»END«: END
```

Sobald Position 33 erreicht ist, bleibt das Programm stehen.

Adresse 10 ($A)

Flagge für LOAD oder VERIFY

In Zelle 10 steht eine 0, wenn geladen wird und eine 1 bei einem VERIFY. Warum das so ist, will ich kurz erläutern:

Die Basic-Routinen für LOAD beziehungsweise für VERIFY sind völlig identisch. Was das Betriebssystem hinterher daraus machen muß, ist natürlich unterschiedlich. Das Basic erspart sich eine doppelte Routine, zeigt aber mit der Flagge in Speicherzelle 10 den Unterschied an.

Erwähnenswert ist noch, daß das Betriebssystem in einer Art Nationalismus seine eigene Flagge aufzieht: Den Unterschied zwischen LOAD und VERIFY speichert es seinerseits in Zelle 147 ($93) ab. Soweit ich es sehen kann, sind Inhalt und Bedeutung beider Speicherzellen völlig identisch.

Ich habe für Sie zwar kein Kochrezept zur Anwendung der LOAD-VERIFY-Flagge in einem Programm vorrätig, möchte Sie aber trotzdem ein bißchen zum Spielen anregen. Um meine Erklärung nachzuvollziehen, tippen Sie bitte direkt LOAD ein. Den Ladevorgang brechen Sie mit der STOP-Taste ab und fragen dann den Inhalt der Zelle 10 ab mit

```
PRINT PEEK (10)
```

Wir erhalten eine 0.

Wiederholen Sie bitte diesen Vorgang, aber mit VERIFY. Wir erhalten jetzt eine 1 — Quod erat demonstrandum.

Wir können auch in die Zelle 10 hineinPOKEn. Die »Wachablösung« zwischen Basic und Betriebssystem unter Hissen der Flagge in Zelle 10 findet beim VC 20 in der Speicherzelle 57705, beim C 64 in 57708 statt. Bevor wir diese Maschinenroutine mit SYS 57705 (SYS 57708) starten, geben wir mit dem Inhalt der Speicherzelle 10 an, ob es ein LOAD oder ein VERIFY sein soll.

Legen Sie ein Band mit Programm in die Datasette. Um ein LOAD zu erzeugen, geben wir direkt ein:

```
POKE 10,0:SYS 57705
(POKE 10,0:SYS 57708)
```

Entsprechend der Anweisung auf dem Bildschirm drücken Sie PLAY und das Auffinden des ersten Programms wird mit LOAD gemeldet. Machen Sie das Ganze noch einmal, diesmal aber POKEn Sie bitte eine 1 in die Zelle 10. Jetzt meldet das Betriebssystem das Auffinden des Programms mit VERIFY.

Wie gesagt, vielleicht fällt Ihnen eine Anwendung dafür ein.

Adresse 11 ($B)

Flagge für den Eingabepuffer/Anzahl der Dimensionen von Zahlenfeldern (Arrays)

Alle Buchstaben und Zeichen, die mit der Tastatur direkt eingetippt werden, kommen in ei-

nen Eingabe-Pufferspeicher. Er beginnt ab Speicherzelle 512 ($200). Sobald die RETURN-Taste gedrückt wird, wandelt eine Routine des Basic-Übersetzers den Text in Codezahlen (Tokens) um. Diese Routine und eine andere, welche die Zeilen eines Programms aneinanderhängt, verwenden die Zelle 11 als Zwischenspeicher.

Sobald die Textumwandlung beendet ist, steht in Zelle 11 eine Zahl, welche die Länge der Token-Zeile angibt.

Die Zelle 11 wird außerdem noch von den Basic-Routinen benützt, die ein Feld (Array) aufbauen oder ein bestimmtes Element in einem Array suchen. Was ein Feld oder Array ist, finden Sie in den Commodore-Handbüchern gut beschrieben.

Diese Routinen also verwenden die Speicherzelle 11, um die Anzahl der verlangten DIMensionen und den für ein neu aufgebautes Feld nötigen Speicherbedarf zu berechnen.

Adresse 12 ($C)

Flagge für Basic-Routinen, die ein Feld (Array) suchen beziehungsweise aufbauen

Diese Speicherzelle wird von den Basic-Routinen als Zwischenspeicher benutzt, die feststellen, ob eine Variable ein Feld (Array) ist, ob das Feld bereits DIMensioniert worden ist, oder ob ein neues Feld die unDIMensionierte Zahl von 11 Elementen hat.

Adresse 13 ($D)

Flagge zur Bestimmung des Datentyps (Zeichenkette/String oder Zahl)

Diese Flagge zeigt den Routinen des Basic-Übersetzers an, ob es sich bei den zur Verarbeitung anstehenden Daten um einen String oder um Zahlenwerte handelt. Zeigt die Flagge 255 ($FF), ist es ein String. Bei 0 handelt es sich um Zahlen. Diese Bestimmung erfolgt jedesmal, wenn eine Variable definiert oder gesucht wird. Diese Flagge kann leider nicht durch ein Basic-Programm abgefragt werden.

Adresse 14 ($E)

Flagge zur Bestimmung des Zahlentyps (Ganze Zahl oder Gleitkommazahl)

Sobald durch die Flagge in der vorherigen Zelle 13 eine Zahl signalisiert wird, steht hier die Zahl 128 ($80) wenn es sich um eine ganze Zahl handelt, während eine 0 die Zahl als Gleitkommazahl identifiziert.

Damit wollen wir ein bißchen experimentieren. Zeile 10 definiert eine Gleitkommazahl, Zeile 20 druckt sie und die Flagge aus Zelle 14 aus.

```
10 A=13.41
20 PRINT A,PEEK (14)
```

Wir erhalten die Zahl 13.41 und als Flagge eine 0.

```
30 B=INT (A)
40 PRINT B,PEEK (14)
```

INT bildet die ganze Zahl von 13.41. Also müßte die Flagge in Zelle 14 auf 128 stehen. Weit gefehlt! Da intern auch die 13 als Gleitkommazahl berechnet wird, erhalten wir immer noch eine 0.

```
50 B%=A
60 PRINT B%,PEEK (14)
```

Erst die Definition der Variablen B als ganze Zahl (mit %) ergibt die Flagge 128.

```
70 D=16*B%
80 PRINT D,PEEK (14)
```

Die Multiplikation einer ganzen Zahl mit der Ganzzahl-Variablen B% fällt in dieselbe Kategorie wie Zeile 30 oben, da die Verarbeitung als Gleitkommazahl erfolgt. Also erhalten wir zu Recht eine 0. Erst wenn D als ganze Zahl (Zeile 90) ausgewiesen wird, steht die Flagge wieder auf 128:

```
90 D%=16*B%
100 PRINT D%, PEEK (14)
```

Adresse 15 ($F)

Flagge bei LIST, Garbage Collection und Textumwandlung

Die Routine des LIST-Befehls muß unterscheiden zwischen Basic-Befehlen und normalem Text. Wenn eine Zeichenkette durch ein »Gänsefüßchen« identifiziert worden ist, wird die Flagge gesetzt, und der Text wird ausgedruckt.

Unter »Garbage Collection« (Müllabfuhr) wird die Routine des Betriebssystems verstanden, welche zu bestimmten Anlässen im Variablenspeicher alle nicht mehr benötigten Strings entfernt, um Platz zu schaffen. Dabei wird eine Flagge in Zelle 15 gesetzt, die anzeigt, daß eine Müllabfuhr bereits stattgefunden hat. Wenn bei der Speicherung eines neuen Strings zuwenig Speicherplatz vorhanden ist, wird bei der Flagge nachgesehen, ob gerade vorher schon durch die Müllabfuhr (Garbage Collection) der Speicher entrümpelt worden ist. Falls das der Fall ist, wird OUT OF MEMORY angezeigt, falls nicht, wird eine Müllabfuhr durchgeführt.

Schließlich wird Zelle 15 auch bei der Umwandlung von Basic-Befehlen in internen Codezahlen (Tokens) eingesetzt.

Adresse 16 ($10)

Flagge zur Anzeige eines Variablen-Feldes oder einer selbstdefinierten Funktion

Im Basic-Übersetzer gibt es eine Routine, die den Speicher absucht, ob es eine Variable mit bestimmten Namen bereits gibt. Wenn diese mit einer Klammer beginnt, wird die Flagge in Zelle 16 gesetzt, um anzuzeigen, daß es sich um eine Array-Variable oder um eine mit DEF FN selbst definierte Funktion handelt.

Adresse 17 ($11)

Flagge für INPUT, GET oder READ

Die Basic-Routinen für INPUT, GET und READ sind zum großen Teil identisch. Um Speicherplatz zu sparen, verwendet der Basic-Übersetzer die identischen Teile nur einmal. Um in die nichtidentischen Teile verzweigen zu können, wird in Zelle 17 angezeigt, um welchen der drei Befehle es sich gerade handelt. Die Flagge steht auf 0 für INPUT, auf 64 ($40) für GET und auf 152 ($98) für READ.

Mit dem folgenden kleinen Programm können wir das leicht nachprüfen:

```
10 DATA 3
20 READ A
30 PRINT PEEK (17)
40 INPUT B
50 PRINT PEEK (17)
80 GET C$:IF C$= ""THEN 60
70 PRINT PEEK (17)
```

Zeile 10 und 20, 40 sowie 60 sind Anwendungen der drei zur Debatte stehenden Basic-Befehle. Nach der Durchführung jedes Befehls wird in den Zeilen 30, 50 und 70 die jeweilige Flagge ausgelesen.

Nach RUN erhalten wir als Resultat der Zeile 20 die Zahl 152, als Resultat von Zeile 30 die INPUT-Aufforderung mit Fragezeichen. Geben Sie irgendeine Zahl und RETURN ein. Wir erhalten so die 0. Die GET-Schleife in Zeile 40 wartet auf einen Tastendruck, dann erhalten wir 64.

Adresse 18 ($12)

Flagge für Vorzeichen des Ergebnisses bei SIN und TAN

Mit dieser Adresse fahren wir das nächste Mal fort.

(Dr. Helmuth Hauck/aa)

Tabelle 1. Was sind Zeiger, Vektoren und Flaggen?

Zeiger, Vektoren und Flaggen

Zeiger und Vektoren sind 2 benachbarte Speicherzellen (Bytes), die eine wichtige Adresse enthalten.
Wir sprechen von einem **Zeiger**, wenn diese Adresse den Beginn von gespeicherten Daten angibt.
Ein **Vektor** kennzeichnet den Beginn eines Maschinenprogramms. (Ich muß zugeben, daß diese Unterscheidung nicht immer scharf angewendet wird beziehungsweise anwendbar ist).
Eine **Flagge** besteht aus einer Zahl, die von einem Programm verwendet wird, um sich das Resultat einer Operation zu merken beziehungsweise für eine spätere Verwendung festzuhalten.

Tabelle 2. Die Zahlendarstellung bei den Commodore-Systemen

Gleitkomma-Zahlen

Für diejenigen Leser, die das Thema der Zahlendarstellung in den Commodore-Handbüchern großzügig übersprungen haben, stelle ich es hier noch einmal vor.

Sie kennen die gängigen vier Zahlentypen:
— ganze Zahlen: 15, 21, 244
— Brüche: $\frac{2}{3}$, $\frac{26}{8}$, $\frac{15}{14}$
— negative Zahlen: —15, —255
— positive Zahlen: 10, 5, 123

Ganze Zahlen bereiten uns und dem Computer keine Probleme.

Bei Brüchen sieht es schon anders aus. Erinnern Sie sich an die Bruchrechnungsstunden in der Schule? Wieviel ist $\frac{51}{52}$ + $\frac{3}{4}$!!

Ohne lang zu überlegen, rechnen wir natürlich um, $\frac{51}{52}$ = 0,9807692 und $\frac{3}{4}$ = 0,75; addiert ist das Resultat 1,7307692 — und schon sind Sie mitten in den Gleitkomma-Zahlen.

Bei obigem Beispiel gleitet allerdings noch nichts. Bei sehr großen oder aber auch sehr kleinen Bruch-Zahlen reicht uns — und einem Computer — nicht der Platz, um sie darzustellen. Die Zahl 0,000000000000000123 sprengt jeden normalen Rahmen.

Daher schreiben wir sie anders. Wir lassen das Komma nach rechts gleiten, bis es die erste Ziffer, die von 0 verschieden ist, findet und für jede Null, die es passiert multiplizieren wir die Zahl mit 10.

Die Zahl oben sieht dann so aus:
0,123 x 10 hoch 15 (eine 1 mit 15 Nullen).

Die Grundzahl vorn heißt »Mantisse«, die 10 mit Hochzahl heißt »Exponent«.

Alle Commodore-Computer verarbeiten intern alle Zahlen in dieser Darstellung, also als Gleitkommazahl (siehe auch Assembler Kurs im 64'er-Magazin, Ausgabe 11)

Assembler ist keine Alchimie

Teil 4

In dieser Folge des Assembler-Kurses lernen Sie die wichtigen Arithmetik-Befehle des Prozessors kennen. Anhand von Beispielen und Übungen können Sie alle Schritte am Computer miterleben. Außerdem wird die Frage geklärt, wie Assembler-Programme in Basic eingebunden werden.

Neun neue Befehle haben wir in der letzten Folge kennengelernt und wir wissen nun, wie unser Computer ganze Zahlen (sogenannte Integers) abspeichert. Zur Erinnerung: Das geschieht im Zweierkomplement-Format. Das Bit 7 einer 8-Bit-Zahl dient dabei als Vorzeichen-Merkmal: Wenn es 0 ist, liegt eine positive Zahl vor, die genauso aussieht, wie wir bislang immer Binärzahlen kannten. Ist das Bit 7 aber eine 1, dann haben wir es mit einer negativen Zahl in der Zweierkomplement-Darstellung zu tun. Wenn wir — wie unser Computer — zur Verarbeitung ganzer Zahlen 16 Bits (also 2 Bytes) verwenden, dann ist eben Bit 15 anstelle von Bit 7 das Vorzeichenbit.

Wenn Sie nun am Ende der letzten Folge ein bißchen mit solchen Zahlen gerechnet haben, konnten Sie sicher feststellen, daß zwar oft das richtige Ergebnis herauskam — aber leider nicht immer. Warum das so ist und was man deswegen noch beim Arbeiten mit Zahlen per Computer beachten muß, soll in dieser Folge dran sein. Damit wir aber nicht nur im vergleichsweise trockenen Zahlendschungel herumirren, sollen Sie heute endlich auch die wichtigsten Befehle des 6502 (beziehungsweise 6510) zur Arithmetik kennenlernen. Außerdem gibt es dazu noch zwei Flaggen gratis und die Branch-Befehle (schon lange überfällig) sollen Ihnen nun vertraut werden. Zunächst aber noch etwas Zahlensalat:

Herr Carry und der V-Mann

Keine Angst, wir sind nicht ins Krimi- oder Agentenmilieu gewechselt! Wir haben es mit zwei Flaggen zu tun, der Carry- und der V-Flagge. »To carry« heißt auf deutsch etwa »tragen«. In der Registeranzeige ist diese Flagge immer mit C gekennzeichnet. Was wird denn hier getragen? Das ergründen wir am besten an einem Beispiel. Dazu rechnen wir mit normalen Binärzahlen (also ohne Rücksicht auf Vorzeichenbits). Wir zählen die Zahlen 128 und 130 zusammen:

128	1000 0000
+ 130	+ 1000 0010
258	(1)0000 0010

Das Ergebnis 258 ist richtig — auch in der Binärdarstellung — nur es paßt nicht mehr in 8 Bits. Ein Bit wurde überTRAGEN in ein extra dafür vorgesehenes Plätzchen: In das Carry-Bit. Jedesmal also, wenn so ein Übertrag in einer Rechenoperation des C 64 stattfindet, zeigt die Carry-Flagge eine 1 (Bild 1).

Je nach Art der von uns programmierten Aufgabe können wir nun dieses Carry-Bit weiterverarbeiten. Es gibt Situationen, in denen man es einfach ignorieren darf (dazu kommen wir gleich noch) oder aber solche, wo man es weiter in der Rechnung verwendet. Schließlich kann es auch noch einen Fehler anzeigen: Dann nämlich, wenn das größte zulässige Ergebnis 1111 1111 sein darf. Natürlich kann das Carry-Bit auch gesetzt werden, wenn man in der Zweierkomplementform rechnet. Die Verhältnisse sind dann aber für ein leicht überschaubares Beispiel des Übertrages zu verwickelt, wie Sie gleich sehen werden.

Wenn wir nämlich mit den Zweierkomplement-Zahlen rechnen, dann interessieren uns auch Fälle wie bei der Addition von 64 und 66:

	64	0100 0000
+	66	+ 0100 0010
(—126)		1000 0010

Das ist offensichtlich falsch. Bei der Addition ist durch das Zusammenzählen der Bits 6 plötzlich Bit 7 gesetzt worden. Da wir es aber mit einer Zweierkomplementzahl zu tun haben, bei der dieses Bit 7 eine negative Zahl anzeigt, folgt ein Fehler. Es ist also von Bedeutung, so einen Überlauf (englisch: 'over-

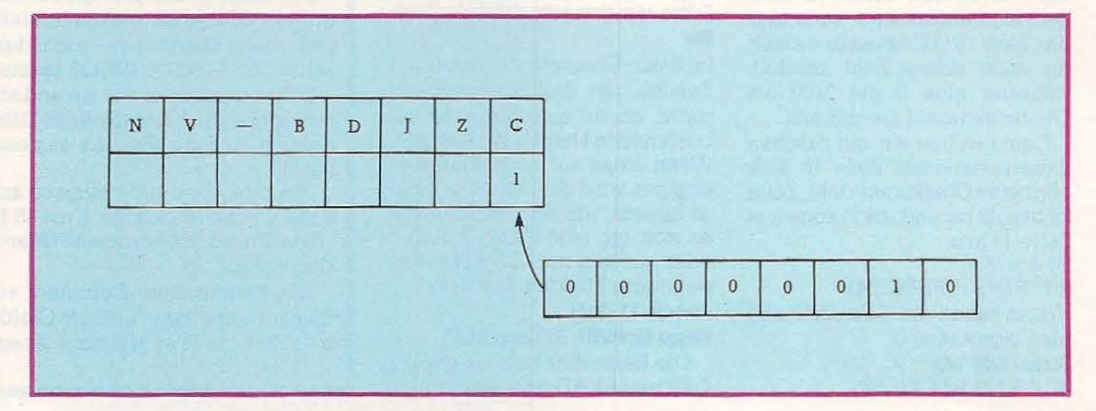


Bild 1. Das Carry-Bit als Bit 8 einer Rechenoperation

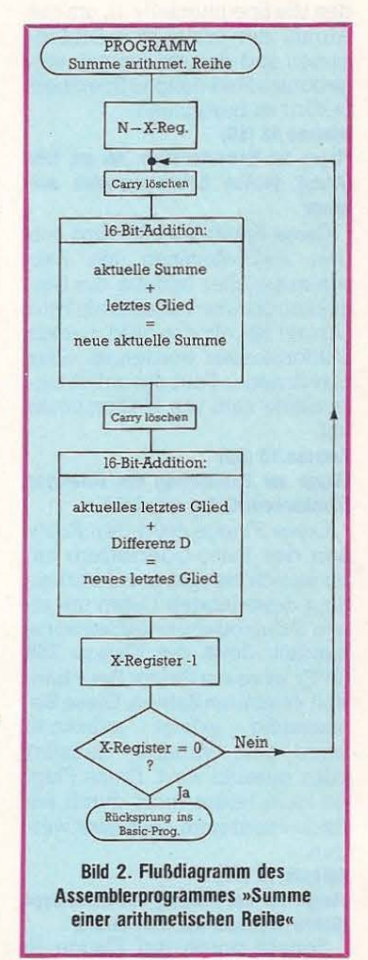


Bild 2. Flußdiagramm des Assemblerprogrammes »Summe einer arithmetischen Reihe«

flow') erkennen zu können um eine entsprechende programmtechnische Reaktion zu starten. Es wird die Überlauf-Flagge V auf 1 gesetzt. Leider ist die Sache aber nicht so einfach, daß sie immer gesetzt würde, wenn von Bit 6 nach Bit 7 ein Übertrag stattfindet. Gesetzt wird diese V-Flagge nur in folgenden zwei Fällen:

1) Es findet ein Übertrag von Bit 6 nach Bit 7 statt, aber kein äußerer Übertrag (wie beim Carry)

2) Es findet kein interner Übertrag von Bit 6 nach Bit 7 statt, aber ein äußerer Übertrag passiert.

Merken kann man sich das am besten so: Immer dann, wenn gewissermaßen das Vorzeichenbit 7 »versehentlich« verändert wurde, wird die V-Flagge auf 1 gesetzt. Das ist ein harter Brocken! Wir sind es ja gewohnt, daß wir uns um diese Dinge beim Computer eigentlich gar nicht mehr kümmern müssen. Außerdem würde das ja erfordern, daß man sich bei allen Operationen vorher überlegen muß, welche Zahlen auftreten können und welche Fehler also durch »versehentliches« Vorzeichenändern passieren können! Genauso ist es — in der Programmierpraxis wird Ihnen aber das ganze Problem nicht mehr so groß vorkommen. Wir wollen uns dieses Zusammenspiel der Überträge von Bit 6 nach Bit 7 und von Bit 7 nach Bit 8 (also ins Carry-Bit) noch anhand einiger Beispiele klarer machen.

Im obigen Beispiel der Addition von 64 und 66 haben wir einen Fall schon behandelt: Es fand ein Übertrag von Bit 6 nach Bit 7 statt, aber kein äußerer Übertrag ins Carry-Bit. Deswegen wurde dann auch die V-Flagge gesetzt. Das Problem läßt sich hier ganz einfach lösen zum Beispiel durch Verwendung von 16-Bit-Zahlen:

	64		0000000001000000
+	66	+	0000000001000010
	130		0000000010000010

Bei 16-Bit-Zahlen ist ja Bit 15 das Vorzeichenbit, welches hier keine Änderung erfährt.

Der andere Fall tritt auf bei der Addition von zwei negativen Zahlen wie —125 und —64:

—125	10000011
— 64	11000000
(+67)	(1)01000011

Auch das ist offensichtlich falsch: Es hat wieder »versehentlich« ein Vorzeichenwechsel stattgefunden. Dies ist also der Fall, wo zwar ein Übertrag ins Carry-Bit stattfand aber kein Übertrag von Bit 6 nach Bit 7. Auch dieses Problem läßt sich durch Verwendung von 16-Bit-Zahlen lösen. Eine kleine Trainingsaufgabe für Sie!

Man kann also sagen: Immer dann, wenn bei 8-Bit-Rechnungen der mittels Zweierkomplementzahlen darstellbare Bereich (127 bis —128) über- oder unterschritten wird, fuhrwerkt man im Vorzeichen-Bit herum und verfälscht das Ergebnis. Dann leuchtet wie eine rote Ampel die Überlauf(V)-Flagge auf und sagt uns, daß wir besser in diesen Fällen mit 16-Bit-Zahlen arbeiten sollten.

Nun noch zum Ignorieren des Carry-Bits, das ich weiter oben erwähnt habe. Bei allen 8-Bit-Rechenoperationen mit Zweierkomplementzahlen kann das Carry-Bit vernachlässigt werden. Zwei Beispiele sollen das wieder illustrieren. Wir addieren +4 und —2:

	+4		00000100
+	—2	+	11111110
	+2		(1)00000010

Das Carry-Bit wird außer acht gelassen. Anderes Beispiel: Wir addieren zwei negative Zahlen, —4 und —6:

	—4		11111010
+	—2	+	11111110
	—6		(1)11111000

Auch hier kann man (sogar: muß man) das Carry-Bit vernachlässigen. Beide Ergebnisse sind richtig.

Nun wissen Sie alles über die Art, wie unser Rechner mit ganzen Zahlen arbeitet. Probieren Sie mal ein paar Aufgaben aus zur Übung.

Wir verlassen jetzt den Zahlendschungel und widmen uns der Praxis.

Der Computer rechnet: ADC, CLC

ADC ist der erste Arithmetik-Befehl des 6502 (und natürlich auch des 6510), den wir kennenlernen. Er bedeutet »add with carry«, also »addiere mit Carry-Bit«. An einem 8-Bit-Beispiel wollen wir uns das mal ansehen. ZAHL1 und ZAHL2 sollen addiert werden. Beide sollen positive 8-Bit-Zahlen sein, die so klein sind, daß kein Überlauf zu erwarten ist. Die ZAHL1 wird in den Akku gegeben:

LDA #ZAHL1

Wenn wir nun den Befehl

ADC #ZAHL2

folgen lassen, sorgt die ALU (arithmetisch-logische Einheit, siehe Folge 1) dafür, daß beide Zahlen addiert werden und das Ergebnis im Akku erscheint. ZAHL1 ist dann vom Ergebnis überschrieben worden. An sich ist damit alles erledigt. Weil wir aber häufig wissen wollen, was denn nun bei der Addition herausgekommen ist, speichern wir den Akku-Inhalt noch irgendwo ab mittels »STA Speicherstelle«. Außerdem war da ja noch die Sache mit dem Carry-Bit. Wir haben oben festgestellt, daß bei einer 8-Bit-Addition kein Carry-Bit berücksichtigt werden soll. Nun gibt es aber eine ganze Menge von Vorgängen in einem Assembler-Programm, die das Carry-Bit beeinflussen. Man kann eigentlich vor einer Addition nie ganz sicher sein, ob es denn nun 1 oder 0 ist. Weil jedoch ADC auch das Carry-Bit mitaddiert, sollte man dafür sorgen, daß es vor dem Zusammenzählen wirklich gelöscht ist. Dazu gibt es den Befehl CLC was die Abkürzung für »clear carry«, also »lösche Carry-Bit« ist. Sei ZAHL1 = 12 und ZAHL2 = 7, dann würde unser vollständiges 8-Bit-Additions-Programmchen also lauten:

```
1200  CLC
1201  LDA #$0C
1203  ADC #$07
1205  STA 1500
```

Sehen wir mal davon ab, daß dieses Programm natürlich unsinnig ist (das kann man ja im Kopf schneller rechnen!), dann erkennen wir: CLC ist ein 1-Byte-Befehl mit impliziter Adressierung, welcher sich nur auf die C-Flagge (also das Carry-Bit) auswirkt. ADC ist in der hier verwendeten Form ein 2-Byte-Befehl und liegt in der »unmittelbar« genannten Adressierung vor. Wie wir oben gesehen haben, kann ADC — je nach Art der Rechnung — auf einige Flaggen wirken: Da wären zunächst natürlich die V-Flagge und die C-Flagge. Dann aber kann beim Auftreten eines gesetzten Bit 7 auch die N-Flagge und beim Überschreiten von $FF eventuell auch die Z-Flagge verändert werden.

Viel interessanter wird unser Mini-Programm schon, wenn man anstelle von

```
1201  LDA #$0C
```

jetzt die absolute Adressierung verwendet, zum Beispiel

```
1201  LDA 1400
```

Weil das ein 3-Byte-Befehl ist, verschiebt sich natürlich der Rest des Programmes um 1 Byte. So kann man immerhin schon zu unterschiedlichen Inhalten von 1400 den gleichen Betrag addieren.

Am interessantesten allerdings ist die Tatsache, daß auch ADC absolut adressierbar ist. Wir können so zum Beispiel den Inhalt der Speicherzelle 1300 zum Inhalt der Zeile 1400 hinzuzählen und dann das Ergebnis in 1500 ablegen:

```
1200  CLC
1201  LDA 1400
1204  ADC 1300
1207  STA 1500
```

Hier ist der ADC-Befehl dann 3 Bytes lang geworden.

Vergessen Sie bitte nicht — das gilt vor allem für die nachfolgenden Rechenoperationen — dann, wenn die Wahrscheinlichkeit besteht, daß der Dezimal-Modus eingeschaltet ist (also die D-Flagge auf 1 gesetzt ist), noch den Befehl CLD vor solche Programme zu stellen.

Solche 8-Bit-Rechnungen kommen recht häufig vor: Wenn man in Schleifen nicht mit mehrfach wiederholten INX (beziehungsweise INY oder INC, DEX, DEY oder DEC) arbeiten will, addiert man eben immer die Sprungweite mittels ADC hinzu. Der Akku kann nicht als Zähler dienen, denn es gibt für ihn keinen Befehl, der dem INX und so weiter vergleichbar wäre, weswegen man ihn — sollte es nötig sein — mittels ADC hochzählt.

Häufiger und in der Praxis bedeutender sind 16-Bit-Rechnungen. Wie Sie sicher noch aus den vorangegangenen Folgen wissen, teilt man so eine 16-Bit-Zahl auf in zwei Bytes (das LSB und das MSB). Nehmem wir für unser nachfolgendes Beispiel wieder an, daß die Zahlen so gebaut sind, daß kein Überlauf zu befürchten ist. ZAHL1 hätten wir vorher in die Speicherstellen 1300 (LSB) und 1301 (MSB) gepackt, ZAHL2 liegt in den Zellen 1400 (LSB) und 1401 (MSB). Zunächst wieder die Vorbereitungsmaßnahmen:

```
1200  CLD
1201  CLC
```

Dabei ist CLD nicht immer nötig, wie schon gesagt. Nun addieren wir zuerst die LSBs:

```
1202  LDA  1300
1205  ADC  1400
1208  STA  1500
```

Ein Überlauf kann hier nicht stattgefunden haben, denn das Vorzeichenbit ist ja im MSB als Bit 15 enthalten, wohl aber kann ein Übertrag stattgefunden haben: Das Ergebnis könnte größer als 255 ($FF) gewesen sein. War das der Fall, dann ist jetzt eine 1 im Carry-Bit. Wir addieren nun die MSBs:

```
120B  LDA  1301
120E  ADC  1401
1211  STA  1501
```

Egal, was im Carry-Bit stand: Es wurde jetzt hinzuaddiert. Das Ergebnis unserer Rechnung steht nun in 1500 (LSB) und 1501 (MSB). Sehen wir uns das ganze nochmal am Zahlenbeispiel an. Wir addieren die Zahlen 2176 (binär: 0000 1000 1000 0000) und 1009 (binär: 0000 0011 1111 0001). Die Speicherinhalte sind dann:

1300	10000000	LSB Zahl1
1301	00001000	MSB
1400	11110001	LSB Zahl2
1401	00000011	MSB

Jetzt addieren wir die LSBs:

1300	10000000
1400	11110001
Carry	0

1500	01110001
Carry:	1

– Nun folgt der zweite Teil der Addition mit den MSBs:

1301	00001000
1401	00000011
Carry:	1
1501	00001100

Damit steht nun das Ergebnis 3185 (binär 0000 1100 0111 0001) säuberlich aufgeteilt in LSB (Speicher 1500) und MSB (Speicher 1501) fest. Das Carry-Bit steht auch nach vollendeter Rechnung noch auf 1, so daß es vor erneuter Addition wieder mit CLC zu löschen ist.

Damit wäre alles über die Addition berichtet. Wie immer in Programmiererkreisen die Empfehlung: Üben, üben,....

Wir wenden uns jetzt der gegenläufigen Operation zu: Der Subtraktion.

Noch mehr Rechnen: SBC, SEC

Daß das Abziehen von Zahlen im Computer durch das Hinzuzählen des Zweierkomplementes geschieht, haben wir mit viel Gehirnschmalzverbrauch schon in vorangegangenen Abschnitten erfahren. Nun sollen Sie die dazu nötigen Befehlsworte des Assemblers kennenlernen. Zunächst einmal ist da SBC. Das heißt »subtract with carry« oder auf deutsch etwa »ziehe unter Berücksichtigung des Carry-Bits ab«. Ebenso wie bei der Addition mit ADC, wirkt das Argument des SBC-Befehls auf den Akku-Inhalt ein — wobei das Ergebnis im Akku landet, diesen also überschreibt. Komplizierter ist hier die Verwendung des Carry-Bits, worauf wir aber nicht detailliert eingehen wollen. (Wen es interessiert: Nachlesen in L.A. Leventhal, »6502 Programmieren in Assembler«, 3. Auflage, München 1983, Seite 3-100). Für uns soll einfach die nicht ganz korrekte Analogie zum »Borgen« bei der Subtraktion ausreichen. Für den Fall, daß ein solches Borgen eintreten muß, sollte auch das dazu nötige Carry-Bit vorhanden sein (also auf 1 gesetzt sein). Wie Sie sicherlich schon erraten haben, heißt SEC »set carry«, also »setze das Carry-Bit« (auf 1).

Merke: Vor einer Addition immer Löschen des Carry-Bits mit CLC, vor einer Subtraktion immer Setzen des Carry-Bits mit SEC!

Zwei Beispiele für die Subtraktion sollen das bisher gesagte erläutern: Zunächst eine 8-Bit-Subtraktion von ZAHL1 (in Speicherzellle 1300) minus ZAHL2 (in Zelle 1400). Ds Ergebnis wird nach 1500 geschrieben:

```
1200   CLD
1201   SEC
1202   LDA 1300
1205   SBC 1400
1208   STA 1500
```

SBC kann — wie hier — absolut adressiert werden, aber auch unmittelbar (also zum Beispiel SBC $40). Der Befehl ist dann im ersten Fall ein 3-, im anderen Fall ein 2-Byte-Befehl. SEC ist ebenso wie vorher schon CLC ein implizit adressierbarer 1-Byte-Befehl.

Das zweite Beispiel ist eine 16-Bit Subtraktion. In den Speichern steht vor dem Aufruf dieser kleinen Routine:

```
1300   ZAHL1 LSB
1301   ZAHL1 MSB
1400   ZAHL2 LSB
1401   ZAHL2 MSB
```

Das Ergebnis soll nach 1500 (LSB) und 1501 (MSB) gebracht werden:

```
1200   CLD
1201   SEC
1202   LDA 1300
1205   SBC 1400
1208   STA 1500
```

Jetzt sind die beiden LSBs voneinander abgezogen und die Differenz abgespeichert als LSB des Erebnisses.

```
120B   LDA 1301
120E   SBC 1401
1211   STA 1501
```

Damit ist die Aufgabe beendet. Auch die MSBs sind subtrahiert und das MSB des Ergebnisses steht in 1501.

SBC beeinflußt die gleichen Flaggen wie der Befehl ADC.

Ein Programmprojekt

Damit die so kennengelernten Arithmetik-Befehle nicht so trocken auf weiter Flur stehen, wollen wir nun ein Programm entwickeln, aus dem zweierlei zu lernen ist:

1) Die Anwendung bisher gelernter Befehle und
2) ein häufig angewendetes Verfahren, Assemblerprogramme in Basic-Programme einzubinden.

Besonders dieser 2. Aspekt scheint noch vielen Lesern unklar zu sein (das zeigen mir Zuschriften). Es gibt eine ganze Reihe von Möglichkeiten, zum Einbau von Assembler-Routinen in Basic-Programme; die werden wir alle nach und nach kennenlernen. Von Ihnen sicherlich schon häufig angewendet wurde der SYS-Befehl (zum Beispiel für SYS 58640 und vorherigem POKE214,Zeile und POKE211, Spalte zum Setzen des Cursors an die Stelle Zeile, Spalte). Damit haben Sie ein Maschinenprogramm aufgerufen, das im System unseres Computers schon enthalten ist. 58640 ist die Startadresse des Programmes und man kann diesen SYS-Befehl eigentlich wie eine Art »GOTO Maschinenprogramm-Startadresse« ansehen. Nichts hindert uns also, auf diese Weise eigene Assembler-Programme anzuspringen! Das Problem liegt nun nur noch darin, wie man Parameter, die unsere Maschinenroutine benötigt, übergeben kann. Eine offensichtliche — aber leider auch relativ langsame — Methode ist das POKEn der Werte im LSB/MSB-Format in die Speicherzellen, aus denen sie sich unser ML-Programm dann abholt. Wir wollen dieses Verfahren nun an einem Programmbeispiel verwenden.

Eine arithmetische Reihe werden viele von Ihnen schon kennen. Wenn man A als erstes Glied, D als Differenz und N als die Anzahl der Glieder bezeichnet, dann ist die Summe einer solchen Reihe:
S=A+(A+D)+(A+2*D)+...
usw....+(A+(N-1)*D)

Ein Beispiel ist die Summe der ersten zehn ganzen Zahlen:
S=1+2+3+4+5+6+7+8+9+10

Hier ist A=1, D=1 und N=10. Daß die Summe S im Beispiel 55 ist, kann man schnell berechnen, was aber, wenn wir wesentlich mehr als nur zehn Glieder haben? Es gibt natürlich auch Formeln zur Berechnung von S. Aber eigentlich ist es ganz reizvoll, ohne solche Formeln den Computer die Summe bilden zu lassen. Wir bauen also ein Programm zur Berechnung der Summe der ersten N ganze Zahlen, wobei N frei gewählt werden kann. Das Ergebnis soll eine 16-Bit-Zahl sein, also nicht größer als 32767. Das beschränkt uns bei N auf Werte von 1 bis 255 (Warum, können Sie ja mal mit dem fertigen Programm ausprobieren). N benötigt also nur 1 Byte Speicherplatz und soll in $1300 abrufbar sein. A soll 1 sein ebenso wie D. Für eventuelle Programmänderungen ist es aber sinnvoll, A und D als 16-Bit-Zahlen aufzubewahren und zwar in $1310/1311 (A in LSB/MSB-Format) und in $1320/1321 (D im gleichen Format). Das Ergebnis soll in $1400/1401 zu finden sein. Das Maschinenprogramm legen wir nach $1200.

Zuerst kümmern wir uns um das Basic-Aufrufprogramm:

```
10  REM **AUFRUF SUMME ARITHMETISCHE REIHE**
20  POKE5120,0:POKE5121,0:REM ERGEBNISSPEICHER AUF
    NULL
30  PRINTCHR$(147)CHR$(17)CHR$(17)
40  INPUT"ANZAHL DER GLIEDER N=";N
50  IFN<1 OR N>255 THEN PRINT CHR$(17)"1<=N
    <=255":GOTO40
60  POKE4864,N:REM EINSPEICHERN VON N
70  POKE4880,1:POKE4881,0:POKE4896,1:
    POKE4897,0:REM EINSPEICHERN VON A UND D
80  SYS4608:REM AUFRUF UNSERES MASCHINENPRO-
    GRAMMES
90  M=PEEK(5121):L=PEEK(5120):REM AUSLESEN DES ER-
    GEBNISSES
100 E=256*M+L:PRINTCHR$(17)CHR$(17)
110 PRINT"DIE SUMME DER ERSTEN "N" GANZEN ZAHLEN
    IST:":PRINTE
120 END
```

Zu diesem Programm gibt es nur noch zu bemerken, daß die Zahlen bei POKE, PEEK oder SYS die Dezimalwerte unserer oben gewählten Adressen sind.

Nun endlich zum Assemblerprogramm. Sehen Sie sich dazu bitte das Flußdiagramm im Bild 2 an.

Wir bereiten den Ablauf vor, indem wir aus $1300 die Anzahl der Glieder ins X-Register laden und zur Vorbereitung der Addition das Carry-Bit löschen. Schalten Sie also bitte den SMON ein und tippen Sie A1200 <RETURN>. Es erscheint die Startadresse 1200. Jetzt können Sie Zeile für Zeile das Assembler-Programm eingeben (nach jeder Zeile ein RETURN, das die nächste Zeilennummer erzeugt):

```
1200   LDX 1300
1203   CLC
```

Die nächsten sechs Zeilen summieren jeweils das neueste Glied zur bis dahin erzeugten Summe. Jetzt zu Beginn ist $1400/1401 noch leer und in $1310/1311 steht noch das Anfangsglied A=1. Später mit Durchlaufen der Schleife, steht in $1400/1401 immer die bis dahin gebildete Summe und in $1310/1311 das letzte Glied der Reihe. Es handelt sich um die oben kennengelernte 16-Bit-Addition:

```
1204   LDA 1400
1207   ADC 1310
120A   STA 1400
```

Das neue LSB ist berechnet und in $1400 geschrieben.

```
1200   LDA 1401
1210   ADC 1311
1213   STA 1401
```

Das war nun noch das neue MSB. Als nächstes berechnen wir das momentan letzte Glied der Reihe durch Addieren von D



zum alten letzten Glied. Das entspricht dem Basic-Befehl A=A+D in einer Schleife. Dies ist eine neue 16-Bit-Addition, weshalb wir wieder CLC vorgeben müssen:

```
1216    CLC
1217    LDA 1310
121A    ADC 1320
121D    STA 1310
```

Das war wieder das LSB. Nun zum MSB:

```
1220    LDA 1311
1223    ADC 1321
1226    STA 1311
```

Wir zählen nun das X-Register um 1 herunter und prüfen, ob es schon Null geworden ist, ob also schon alle N-Glieder summiert worden sind:

```
1229    DEX
122A    BNE 1203
```

Wenn noch nicht alle Glieder berechnet und summiert sind, kehren wir an den Schleifenanfang zurück. Ansonsten springen wir zurück ins Basic-Aufrufprogramm:

```
122C    RTS
```

Wenn Sie beide Programme eingetippt haben, dann speichern Sie sie vorsichtshalber ab (das Assemblerprogramm mit dem S-Befehl des SMON). Beim neuen Einladen brauchen Sie den SMON nicht mehr. Nach dem Laden unseres Maschinenprogrammes (mit ,8,1 bei Diskette oder ,1,1 bei Kassette) geben Sie NEW <RETURN> ein, damit die Zeiger vor dem Einladen des Basic-Programmes wieder auf Normalwerte gesetzt werden. Zwischen dem dann eingeladenen Basic-Programm und unserer Assembler-Routine ist genug Platz. Sollten Sie aber irgendwann mal das Basic-Programm vergrößern, schützen Sie bitte unseren Bereich ab $1200.

Unser Assembler-Beispiel ist so aufgebaut, daß auch A und D variabel gehalten sind. Sie müßten dann nur noch Eingabemöglichkeiten im Basic-Programm schaffen und anstelle der Werte 1 oder 0 in Zeile 70 die LSBs und MSBs der von Ihnen eingegebenen Größen A und D einPOKEn. Auf diese Weise sind dann beliebige ganzzahlige, arithmetische Reihen berechenbar, wie zum Beispiel S=7+10+13+16+... und so weiter. Das überlasse ich Ihrer Basic-Programmierfertigkeit. Nur eines noch: Sie müssen darauf achten, daß die Summe S nicht größer als 32767 wird. Ihrer Phantasie sind — wie immer in diesem Metier — keine Grenzen gesetzt. Sie könnten sich ja mal überlegen, wie man größere Summen zulassen kann (wer sagt denn, daß wir Zahlen immer nur in 2 Bytes darstellen dürfen?). Oder Sie könnten sich überlegen, welches eindeutige Merkmal auftritt, sobald der Maximalwert überschritten wird (ein Tip: Lesen Sie doch mal den Abschnitt über die V-Flagge nach).

Die Branch-Befehle

Der 6502 (und auch der damit identische 6510) kennt acht bedingte Verzweigungen, von denen wir bisher BNE schon verwendet haben. Alle diese Branch-Befehle (von branch = verzweigen) prüfen Flaggen des Statusregisters.

BNE und BEQ beziehen sich auf die Z-Flagge, die anzeigt,ob im Verlauf der letzten Operation eine Null aufgetreten ist. Ist das der Fall, steht in der Z-Flagge eine 1. BNE verzweigt zur angegebenen Adresse, wenn in der Z-Flagge eine 0 enthalten ist. BEQ (branch if equal zero = verzweige, wenn gleich Null) tut das dann, wenn die Z-Flagge auf 1 gesetzt ist. Da muß man etwas aufpassen, daß man sich nicht vertut!

BCC und BCS haben ihre Aufmerksamkeit auf die C-Flagge, also das Carry-Bit gerichtet. BCC kommt vom englischen »branch if carry clear«, was heißt: »verzweige, wenn das Carry-Bit gelöscht ist«. Ein gesetztes Carry-Bit (also Inhalt=1) veranlaßt BCS (»branch if carry set« = verzweige, wenn das Carry-Bit gesetzt ist) zum Sprung an die angegebene Adresse.

Diese vier bedingten Verzweigungen sind an sich die bedeutsamsten und am häufigsten verwendeten Branch-Befehle. Man kann wohl getrost sagen, daß über 90% der von Programmierern verwendeten bedingten Sprünge, damit absolviert werden. R. Mansfield warnt sogar ausdrücklich in seinem Buch »Machine language for beginners«, einem in den USA sehr verbreitetem Werk, vor der Verwendung der Befehle BPL und BMI!

Dafür liegt absolut kein einsehbarer Grund vor. Viele programmtechnischen Aufgabenstellungen lassen sich elegant und leicht mit BPL, BMI, BVS und BVC lösen. Man muß nur wissen, wie sie funktionieren und — da liegt vermutlich der Hund begraben — man muß auch die Art kennen, wie Zahlen vom Computer behandelt werden. Genau das aber wissen wir und deswegen sollten wir diese Kenntnis für uns auch nutzen. Also ohne Scheu heran an die verfehmten Befehle!

BMI und BPL (branch on minus = verzweige, wenn negativ und branch on plus = verzweige, wenn positiv) hängen mit der Negativ-Flagge N zusammen. Das Rätsel dieser Flagge konnte in den vorangegangenen Folgen gelöst werden: Immer dann, wenn bei einer Operation eine Zahl auftrat, deren Bit 7 eine 1 war, wurde die N-Flagge auf 1 gesetzt. Wir wissen jetzt, daß dieses Bit bei 8-Bit-Zahlen das Vorzeichenbit ist. Bit 7 sagte uns bei einer 1, daß eine negative Zahl im Zweierkomplement-Format vorliegt oder aber überhaupt ein Speicherzelleninhalt vorhanden ist, der größer als 0111 1111 = 127 ist. BMI führt zum Sprung in diesem Fall, weil die N-Flagge auf 1 steht. Andernfalls führt BPL zur Verzweigung.

Ebenso einfach sind BVS und BVC zu erklären: Sie beziehen sich auf die V-Flagge, unsere rote Ampel, die Überlauf bei Rechenoperationen anzeigt. Kann es was bequemeres geben zur Behandlung solcher Fehlrechnungen als ein »branch on overflow set« = »verzweige, falls die Überlauf-Flagge gesetzt (=1) ist« mit BVS? Oder anders herum bei BVC »branch on overflow clear« = »verzweige bei freier Überlauf-Flagge«. Wenn man — wie Sie jetzt nach dieser Folge — weiß, unter welchen Umständen diese V-Flagge auf 1 gesetzt wird, sollte man ohne Skrupel BVS und BVC ausgiebig benutzen. Man könnte damit zum Beispiel programmieren, daß die Rechengenauigkeit automatisch von 16-Bit auf 24- oder 32- (oder wie es gerade beliebt) Bit gesteigert wird, ohne daß man sich bei jeder Programmaufgabe Gedanken über das größtmögliche Ergebnis machen muß. Dazu aber ein andermal mehr.

Alle hier vorgestellten Branch-Befehle sind ebenso wie BNE 2-Byte-Befehle, was an der speziellen Art der Adressierung liegt: Der relativen Adressierung.

Eigentlich hatte ich Ihnen ja versprochen, diese relative Adressierung zusammen mit den Branch-Befehlen zu erklären. Ich werde ihr aber lieber einen eigenen Abschnitt widmen, weil's zum genauen Verständnis doch etwas mehr an Aufwand braucht. Die nächste Folge fängt dann damit an, abgemacht?

Wie die anderen Folgen auch, soll auch diese hier noch mit einer Tabelle enden, in der die neu gelernten Befehle mit Zubehör gezeigt sind.

(Heimo Ponnath/gk)

Befehlswort	Adressierung	Byteanzahl	Code		Dauer in Taktzyklen	Beeinflussung von Flaggen
ADC	unmittelbar	2	69	105	2	N,V,Z,C
	absolut	3	6D	109	4	
CLC	implizit	1	18	24	2	0→C
SBC	unmittelbar	2	E9	233	2	N,V,Z,C
	absolut	3	ED	237	4	
SEC	implizit	1	38	56	2	1→C
BEQ	relativ	2	F0		2	keine Änderung
BCC	relativ	2	90		2	keine Änderung
BCS	relativ	2	B0		2	keinc Änderung
BMI	relativ	2	30		2	keine Änderung
BPL	relativ	2	10		2	keine Änderung
BVC	relativ	2	50		2	keine Änderung
BVS	relativ	2	70		2	keine Änderung

(Branch-Befehle, Dauer:) +1 bei Verzweigung, +2 bei Überschreiten einer Seitengrenze

Tabelle: Die 11 neuen Befehle



In den letzten beiden Folgen sind die Möglichkeiten, die die Basic bietet, ausgeschöpft worden. Heute soll deswegen die Bedienung der Floppy in Maschinensprache im Vordergrund stehen.

Wenn wir in dieser Folge von Routinen sprechen, die im Betriebssystem stehen, so werden wir die in Tabelle 1 dargestellten Kürzel verwenden, die Sie übrigens auch in Editorprogrammen gut benutzen können.

FILPAR und FILNAM

Bei OPEN, LOAD und ähnlichen Befehlen müssen Sie entsprechenden Routinen mitteilen, welches File Sie wo öffnen wollen. Um Ihnen eine »Herumwurstelei« in der Zeropage zu ersparen, wo Sie die einzelnen Angaben von Hand setzen müßten, hat das Betriebssystem zwei entsprechende Routinen implementiert. FILPAR setzt für Sie die einzelnen Fileparameter. Diese müssen der Routine in den Prozessorregistern übergeben werden:

— Filenummer (Akku)
— Geräteadresse (X-Register)
— Sekundäradresse (Y-Register)

Ein Beispiel:
Sie wollen für ein File mit der Nummer 1, der Geräteadresse 8 und der Sekundäradresse 15 (Kommandokanal der Floppy) die entsprechenden Fileparameter setzen:

```
LDA   #$01      ; Filenummer 1
LDX   #$08      ; Geräteadresse 8
LDY   #$6F      ; Sekundäradresse + $60
JSR   #FILPAR   ; Fileparameter setzen
```

Wie Sie sehen, muß zu der betreffenden Sekundäradresse ein Wert von $60 addiert werden.

Aber in vielen Fällen müssen Sie ja auch einen Filenamen angeben. Dazu dient die FILNAM-Routine. Hier erfolgt die Parameterübergabe:

— Länge des Filenamens (Akku)
— Adresse LO des Namens (X-Register)
— Adresse HI des Namens (Y-Register)

Und wieder ein Beispiel. Um das Directory-File mit dem Namen »$« anzusprechen, geben Sie bitte folgende Befehle ein:

```
LDA   #$24     ; Code für '$' in Akku
STA   $FF      ; und abspeichern
LDA   #$01     ; Länge des Filenamens
LDX   #FF      ; Adresse LO
LDY   #$00     ; Adresse HI
JSR   FILNAM   ; übergeben
```

Sie müssen also wissen, wo der Filename im Speicher steht und wie lang er ist. Dies ist aber im allgemeinen kein Problem. Auf die gleiche Weise können Sie der Floppy über den Kommandokanal auch Befehle senden, wie Sie in der letzten Folge vorgestellt wurden. Das entspräche der Basic-Sequenz:
OPEN x, 8, 15, "befehl"

Natürlich können Sie auch alle Parameter von Hand setzen, beziehungsweise noch einmal lesen. Wo sich die einzelnen Parameter in der Zero-Page nach Ausführung dieser und der anderen Routinen befinden, ist in Tabelle 2 angegeben.

OPEN und CLOSE

Nachdem wir alle Fileparameter und den Filenamen übergeben haben, können wir die OPEN-Routine mit JSR OPEN aufrufen. Schon ist das entsprechende File geöffnet. Zu beachten wäre folgendes: Es können im Computer niemals mehr als 10 Files gleichzeitig geöffnet sein!

Die CLOSE-Routine arbeitet analog zu OPEN, mit der Ausnahme, daß nur die Filenummer übergeben werden muß. Geräteadresse und Sekundäradresse sucht sich der C 64 aus einer Tabelle heraus, auf die wir später noch zu sprechen kommen:

```
LDA   #$01    ; Filenummer
JSR   CLOSE     in Akku
```

Der Filename wird beim Schließen überhaupt nicht mehr benötigt.

LISTEN und UNLIST, TALK und UNTALK

Nach dem Öffnen eines Files kann die Datenübertragung noch nicht beginnen. Sie müssen dem entsprechenden Gerät zuerst mitteilen, ob es senden oder empfangen soll.

Bestes Beispiel ist wieder der Kommandokanal. Über diesen kann die Floppy sowohl Befehle empfangen, als auch Fehlermeldungen senden. Um ein Gerät zum Empfangen zu veranlassen, verwenden wir die Routine LISTEN. Das hat nichts mit dem Basic-Befehl LIST zu tun, sondern kommt vom englischen Wort für »Hören«. Beim Aufruf vom LISTEN ist das angesprochene Gerät auf Empfang und der Computer auf Senden eingestellt.

Wichtig ist, daß der Akku beim Aufruf die Geräteadresse enthält. Dies gilt für alle vier hier beschriebenen Routinen. Wenn Sie mit dem Senden der Daten fertig sind, sollten Sie ein UNLISTEN zum entsprechenden Gerät schicken, um dieses wieder freizugeben. Dies geschieht mit Hilfe der UNLIST-Routine. Analog verhält es sich mit den Routinen TALK und UNTALK. Sie veranlassen das angesprochene Gerät, Daten zu senden, beziehungsweise mit dem Senden aufzuhören und wieder in den Wartezustand zurückzukehren.

SECTLK und SECLST

Die beiden Routinen SECTLK und SECLST sind ebenfalls sehr wichtig für die Datenübertra-

Auflistung aller verwendeten ROM-Routinen

Kürzel	Adresse	Kürzel	Adresse
FILPAR	$FFBA	SECTLK	$FF96
FILNAM	$FFBD	SECLST	$FF93
OPEN	$FFC0	IECOUT	$FFA8
CLOSE	$FFC3	IECIN	$FFA5
LISTEN	$FFB1	FILTAB	$F30F
UNLIST	$FFAE	FILSET	$F31F
TALK	$FFB4	LOAD	$FFD5
UNTALK	$FFAB	SAVE	$FFD8
		BASOUT	$FFD2
		CLALL	$FFE7

Tabelle 1. Die im Artikel erwähnten Betriebssystemroutinen

Adresse	Bedeutung
$90	Status-Flag
$93	Flag für LOAD/VERIFY
$98	Anzahl der offenen Files
$99	Eingabegerät für BASIN
$9A	Ausgabegerät für BASOUT
$B7	Länge Filename
$B8	aktive Filenummer
$B9	Sekundäradresse
$BA	Geräteadresse
$BB/BC	Zeiger auf Filenamen

Wichtige Zeropageadressen

Tabelle 2. Dies sind die Zeropageadressen unter denen die aktuellen Fileparameter gespeichert werden

Byte an das aktuelle Gerät; IECIN empfängt eines und legt es im Akku ab.

Busfehlerbehandlung

Bei aller Sorgfalt; Fehler können immer auftreten, so auch beim Busbetrieb. Um einen in einer Busroutine aufgetretenen Fehler zu signalisieren, verwendet das Betriebssystem das Carry-Flag. Generell gilt: Ist das Carry-Flag gesetzt, so ist etwas nicht in Ordnung, und wir sollten das Statusbyte überprüfen. Dieses Statusbyte steht in der Speicherstelle $90. Immer wenn es ungleich Null ist, liegt irgendein Sonderfall vor. Jedes Bit des Statusbytes hat eine andere Funktion; Tabelle 3 zeigt diese Belegung. Ist zum Beispiel das Bit 7 gesetzt, so ist das angesprochene Gerät entweder nicht vorhanden oder abgeschaltet. In Basic bekämen wir in einem solchen Fall die Meldung »DEVICE NOT PRESENT ERROR«. Interessant ist für uns noch das Bit 6. Ist es gesetzt, so bedeutet das, daß das letzte Byte der angeforderten Informationen übertragen wurde. Dies können wir uns auch in Basic zunutze machen, um beispielsweise die Fehlermeldung der Floppy auszulesen:

```
10 OPEN1,8,15
20 GET#1, A$: PRINTA$;: IF ST<>64 THEN 20
30 CLOSE1
```

Wie Sie an diesem Beispiel sehen, ist der Inhalt der Speicherstelle $90 in der Variablen ST ▶

```
(In $FA/$FB muß die Adresse, in $FC
die Länge des Befehlsstrings
stehen.)
     LDA #$01    ; Filenummer
     LDX #$08    ; Gerätenummer
     LDY #$6F    ; Sekundäradresse
     JSR FILPAR  ; setzen.
     LDA #$00    ; Länge Filename =0
     JSR FILNAM  ; da kein Filename.
     JSR OPEN    ; File öffnen
     LDA #$08    ; Geräteadresse
     JSR LISTEN  ; auf Empfangen
     LDA #$6F    ; Sekundäradresse
     JSR SECLST  ; senden.
     LDY #$00    ; Zähler auf Null
  1  LDA (FA),Y  ; Befehlsbyte laden
     JSR IECOUT  ; und übertragen
     INY         ; Zähler erhöhen
     CMP $FC     ; Befehlslänge
     BNE 1       ; noch ein Byte ?
     LDA #$08    ; Geräteadresse
     JSR UNLIST  ; Sendung beenden
     LDA #$01    ; Filenummer
     JSR CLOSE   ; Schliessen
     RTS
```

Listing 1. So können Befehlsstrings an die Floppy gesendet werden

```
Prinzip des Lesens des Fehlerkanals
mit Ausgabe auf dem Bildschirm.
     LDA #$00    ; Zurücksetzen des
     STA $90     ; Status-Flags
     LDA #$01    ; Filenummer
     LDX #$08    ; Geräteadresse
     LDY #$6F    ; Sekundäradresse
     JSR FILPAR  ; setzen.
     LDA #$00    ; Länge Filename =0
     JSR FILNAM  ; setzen.
     JSR OPEN    ; File öffnen
     LDA #$08    ; Geräteadresse auf
     JSR TALK    ; Senden schalten
     LDA #$6F    ; Sekundäradresse
     JSR SECTLK  ; übertragen
  1  JSR IECIN   ; Byte empfangen
     JSR BASOUT  ; und ausgeben
     BIT $90     ; Bit 6 Status =0?
     BVC 1       ; dann noch ein Byte
     LDA #$08    ; Geräteadresse
     JSR UNTALK  ; Sendung beenden
     LDA #$01    ; Filenummer
     JSR CLOSE   ; und schliessen
     RTS
```

Listing 2. So läßt sich der Fehlerkanal auslesen und anzeigen

gung. Denn obwohl wir beim OPEN-Befehl eine Sekundäradresse angeben, muß diese bei jeder weiteren Übertragung nochmals an das aktuelle Gerät gesendet werden. Dies hat zwei Gründe: Einerseits können Sie ja mehrere Floppykanäle gleichzeitig geöffnet halten. Damit die Floppy nun weiß, für welchen Kanal der nächste Schwung von Daten bestimmt ist, beziehungsweise, welcher Kanal senden soll, muß nach dem Aufruf von TALK SECTLK, beziehungsweise nach dem Aufruf von LISTEN SECLST durchgeführt werden. Außerdem merkt sich der Computer zwar die angegebene Sekundäradresse, sendet sie aber nicht. Dies hat praktische Gründe, wie wir noch bei den LOAD/SAVE-Routinen sehen werden. SECTLK und SECLST benötigen die jeweilige Sekundäradresse +$60 im Akku. Diese kann, wie in unseren Beispielen, direkt geladen oder aber auch der entsprechenden Zero-Page-Adresse entnommen werden.

IECOUT und IECIN

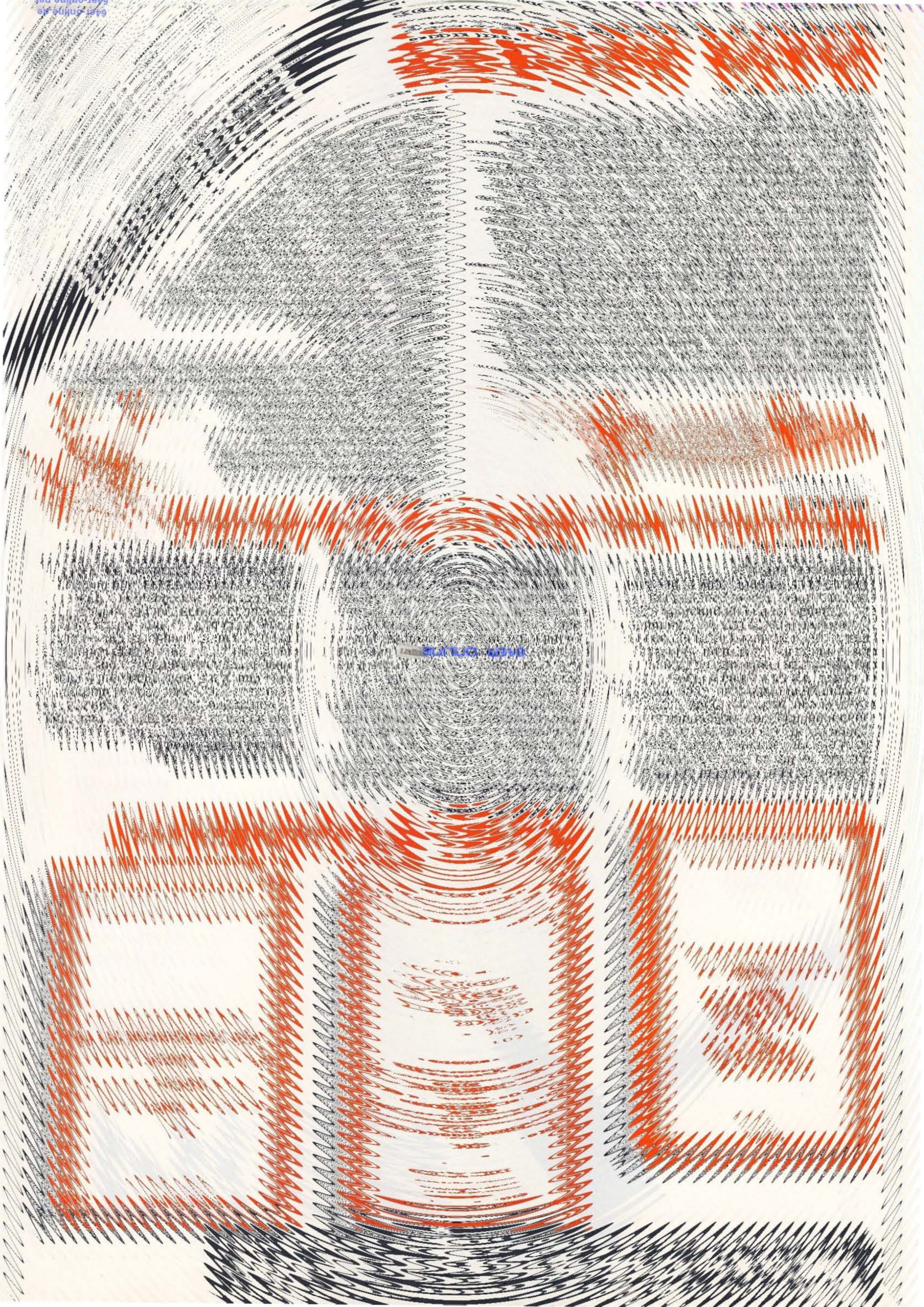
Nachdem wir nun endlich alle Vorbereitungen getroffen haben, können wir munter Bytes von der Floppy zum Computer und umgekehrt übertragen. Dies ist mit den ROM-Routinen denkbar einfach. IECOUT überträgt das im Akku befindliche

Das Status-Flag

Bit	Bedeutung wenn gesetzt
1	Fehler (Zeitüberschreitung) bei IEC-Eingabe
2	Fehler (Zeitüberschreitung) bei IEC-Ausgabe
3-5	nur für Kassettenbetrieb
6	Übertragung ist beendet
7	Gerät meldet sich nicht

Tabelle 3. Für uns wichtige Bits im Statusflag





```
Prinzip des Ladens von Programmen.
LDX #$08    ;  Geräteadresse
LDY #$00    ;  Sekundäradresse für
               relativ laden
JSR FILPAR ;   und setzen.
LDX # (Filename LO-Byte)
LDY # (Filename HI-Byte)
LDA # (Filename Länge)
JSR FILNAM
LDA #$00    ;  LOAD-Flag
LDX # (Startadresse LO-Byte)
LDY # (Startadresse HI-Byte)
JSR LOAD
RTS
```

Listing 3. Das Laden von Programmen an beliebige Adressen

enthalten. Vor jeder neuen Datenübertragung sollten Sie darauf achten, daß das Statusbyte gelöscht wird, da sonst irrtümlich Fehler festgestellt werden könnten. Zur Verdeutlichung des bisher Gesagten dienen die Listings 1 und 2, die jedoch nur Anhaltspunkte geben sollen. Sie sind weder perfekt noch eintippfertig und sollten auf den jeweiligen Bedarf abgestimmt werden.

Bearbeiten mehrerer Files

Sie werden festgestellt haben, daß wir bisher immer nur mit einem einzigen File gearbeitet haben. Was aber, wenn Sie gleichzeitig zwei Files offen halten müssen, zum Beispiel, um einen Block von Diskette zu lesen. Sie erinnern sich ja, daß wir dazu sowohl den Kommandokanal als auch einen Übertragungskanal benötigen. Wir könnten zwar jeweils, wenn wir den Kanal wechseln wollen, mit CLOSE den alten schließen und mit OPEN den neuen öffnen, aber es geht auch einfacher.

Voraussetzung ist, daß alle benötigten Files schon geöffnet sind. Dann kann mit Hilfe einer, schon erwähnten, Filetabelle zwischen — bis zu 10 — Files beliebig umgeschaltet werden. Diesen Zweck erfüllen die Routinen FILTAB und FILSET.

FILTAB benötigt im Akku die Nummer des Files, auf das Sie umschalten wollen. Die Routine sucht dann in der Filetabelle nach den entsprechenden anderen Parametern. Tritt hier ein Fehler auf, weil das File noch gar nicht geöffnet wurde, so wird das Zero-Flag gelöscht und es kann mit BNE auf einen Fehler überprüft werden.

FILSET schreibt dann die gefundenen Parameter in die entsprechenden Zero-Page-Adressen. Die komplette Routine zum Umschalten auf das File x lautet also:

```
LDA  #$ xx   ; Nummer des Files
JSR  FILTAB  ; Durchsuchen der Tabelle
BNE  ERROR   ; Fehler ?
JSR  FILSET  ; Parameter setzen
```

Die ERROR-Routine müssen Sie natürlich noch selbst schreiben. Danach ist das angewählte File zum aktuellen File geworden. Alle LISTEN, TALK und so weiter, beziehen sich jetzt auf dieses neue File.

In den Zero-Page-Adressen aus Tabelle 2 stehen nun die für dieses File aktuellen Parameter, da sie aus der großen Filetabelle automatisch übertragen werden. Eine Ausnahme bildet hier der Filename, da er nur beim Öffnen des Files benötigt wird.

Diese große Filetabelle befindet sich übrigens an den Speicherstellen $0259 bis $0276.

Denken Sie immer daran, vor einem erneuten Umschalten UNLIST oder UNTALK aufzurufen. CLOSE braucht dagegen erst aufgerufen zu werden, wenn die Bearbeitung eines Files völlig abgeschlossen ist.

LOAD und SAVE

Prinzipiell könnten Sie mit dem bisher Erwähnten auch schon Programme laden und speichern, allerdings nur sehr mühselig. Da unser Computer das aber schon von selbst beherrscht, geben wir ihm gern diese Arbeit ab.

Betrachten wir zunächst die LOAD-Routine. Auch hier muß wieder eine Vielzahl an Parametern übergeben werden. Mit FILPAR werden Gerätenummer und Sekundäradresse gesetzt. Eine Filenummer braucht nicht gesetzt zu werden. Für die Sekundäradresse gilt folgendes:

Ist sie gleich Null, so wird das Programm an eine, von Ihnen festgelegte, Speicherstelle geladen. Ist sie gleich Eins, so wird das Programm an die Speicherstelle geladen, an der es bei SAVE stand. Der erste Modus wird vom Betriebssystem ausgenutzt, um Programme ab $0800 zu laden, wenn beim LOAD-Befehl keine Sekundäradresse angegeben wird. Prinzipiell kann aber an jede beliebige Adresse geladen werden! Der Filename wird, wie gewohnt, mit FILNAM gesetzt. Vor dem Aufruf der LOAD-Routine treten zwei, uns neue, Parameter hinzu, die wie folgt übergeben werden:
LOAD/VERIFY Flag (Akku)
Ladeadresse LO (X-Register)
Ladeadresse HI (Y-Register)

Steht beim Aufruf der Routine im Akku 0, so wird geladen. Steht dort hingegen eine 1, so wird ein VERIFY durchgeführt.

Die Startadresse in den X/Y-Registern wird nur beachtet, wenn die Sekundäradresse gleich Null ist. Alles übrige erledigt die LOAD-Routine, und Sie brauchen nur noch deren Ende abzuwarten. Zur Sekundäradresse wäre noch folgendes zu bemerken:

```
Prinzip des Speicherns von
Bereichen.

LDX #$08    ;  Geräteadresse
JSR FILPAR ;   setzen
LDX # (Filename LO-Byte)
LDY # (Filename HI-Byte)
LDA # (Filename Länge)
JSR FILNAM ;   setzen
LDX # (Startadresse LO-Byte)
LDY # (Startadresse HI-Byte)
STX  $FB    ;  zwischenspeichern
STY  $FC
LDA #$FB    ;  Pointer auf Startadr.
LDX # (Endadresse +1 LO-Byte)
LDY # (Endadresse +1 HI-Byte)
JSR SAVE
RTS
```

Listing 4. Und so funktioniert das Abspeichern

Egal, was Sie für eine Adresse angeben, zur Floppy wird immer nur 0 gesendet. Wie Sie schon wissen, ist diese Sekundäradresse floppyintern für den LOAD-Befehl reserviert und darf nicht ohne weiteres bei OPEN-Befehlen verwendet werden. Nach Beendigung der LOAD-Routine wird im X und Y-Register die Endadresse des Programms übergeben.

Die SAVE-Routine hat eine etwas kompliziertere Parameterübergabe. FILPAR braucht nur mit der Gerätenummer im X-Register aufgerufen zu werden, da weder Sekundäradresse noch Filenummer benötigt werden. Das Setzen des Filnamens erfolgt normal über FILNAM.

Übergeben werden müssen nun noch Anfangsadresse und Endadresse+1 des zu speichernden Bereichs. Die Anfangsadressen müssen Sie irgendwo in der Zero-Page in der Reihenfolge LO/HI ablegen. Empfehlenswert wären die Adressen $FB/FC, da diese nicht vom Betriebssystem oder Basic benutzt werden. Im Akku muß dann die Adresse des LO-Byte übergeben werden; wenn Sie die Adresse also unter $FB/FC speichern, muß im Akku $FB stehen.

Die Endadresse übergeben Sie wie folgt:
LO-Byte im X- und HI-Byte im Y-Register. Es muß immer 1 zur Engadresse addiert werden, da sonst das letzte Byte des Programms nicht abgespeichert wird. Danach kann die Routine SAVE aufgerufen werden. Wieder haben wir für Sie zur Verdeutlichung zwei Listings: Listing 3 zeigt, wie man ein Programm an eine beliebige Adresse lädt; Listing 4 wie man einen beliebigen Bereich auf Diskette speichert. Erwähnenswert ist noch die Routine CLALL, die **alle** Files im Computer schließt; die Kanäle in der Floppy bleiben davon jedoch unberührt. Hier müssen Sie also sorgfältig mit CLOSE arbeiten, da Sie sonst Daten verlieren können.

Nachdem wie Sie nun mit Theorie überschwemmt haben, sollen Sie sogleich in den Genuß Ihrer neuen Kenntnisse kommen. Haben Sie schon einmal etwas

```
.
., 033C 20 79 00 JSR $0079
., 033F F0 43    BEQ $0384
., 0341 20 E7 FF JSR $FFE7
., 0344 A0 00    LDY #$00
., 0346 B9 A5 03 LDA $03A5,Y
., 0349 F0 06    BEQ $0351
., 034B 20 D2 FF JSR $FFD2
., 034E C8       INY
., 034F D0 F5    BNE $0346
., 0351 20 54 E2 JSR $E254
., 0354 20 C1 F5 JSR $F5C1
., 0357 A6 B7    LDX $B7
., 0359 F0 66    BEQ $03C1
., 035B A9 01    LDA #$01
., 035D A2 08    LDX #$08
., 035F A0 02    LDY #$02
., 0361 20 BA FF JSR $FFBA
., 0364 20 C0 FF JSR $FFC0
., 0367 A9 04    LDA #$04
., 0369 20 B1 FF JSR $FFB1
., 036C 20 BE ED JSR $EDBE
., 036F A2 01    LDX #$01
., 0371 20 C6 FF JSR $FFC6
., 0374 20 BE ED JSR $EDBE
., 0377 20 85 EE JSR $EE85
., 037A 20 97 EE JSR $EE97
., 037D A9 00    LDA #$00
., 037F 85 99    STA $99
., 0381 85 98    STA $98
., 0383 60       RTS
., 0384 A9 01    LDA #$01
., 0386 85 98    STA $98
., 0388 20 AE FF JSR $FFAE
., 038B 20 AB FF JSR $FFAB
., 038E A9 01    LDA #$01
., 0390 20 C3 FF JSR $FFC3
., 0393 A0 00    LDY #$00
., 0395 B9 AF 03 LDA $03AF,Y
., 0398 F0 06    BEQ $03A0
., 039A 20 D2 FF JSR $FFD2
., 039D C8       INY
., 039E D0 F5    BNE $0395
., 03A0 A9 00    LDA #$00
., 03A2 4C 74 A4 JMP $A474
.D 03A5
.?
.
.'03A5 53 50 4F 4F 4C 49 4E 47
.'03AD 20 00 45 4E 44 20 4F 46
.'03B5 20 53 50 4F 4F 4C 49 4E
.'03BD 47 8D 00 00 4C 08 AF 00
.I 03C5
.?
.
., 03C1 4C 08 AF JMP $AF08
.D 03C4
.?
.
READY.
```

Listing 5. Mit diesem Programm können Sie ein Floppy-Drucker-Spooling durchführen. Näheres im Text.

von Spooling gehört? Nein? Macht nichts, wir werden uns mit dieser Technik nämlich jetzt auseinandersetzen, und Sie werden dabei die Vorzüge dieser Möglichkeit genießen lernen.

Spooling? Was ist das?

Unter dem Begriff Spooling verbirgt sich eigentlich eine ganz einfache Technik, die jedoch enorme Vorteile besitzt: Es handelt sich um das Drucken direkt von Diskette. Haben Sie nicht auch schon öfters versucht, ein meterlanges Listing auf Papier zu bringen und den Drucker dabei mit wütenden Blicken zu größerer Eile aufgefordert, weil Sie nämlich unter Zeitdruck standen und sich bei der Arbeit keine Verzögerung erlauben konnten? Dann ist Spooling genau das Richtige für Sie. Bei dieser Methode wird ein Listing, das ausgedruckt werden soll, auf Diskette gebracht. Danach starten Sie ein Spooling-Programm und siehe da; der Drucker beginnt Ihr Listing auf Papier zu bringen, und der Computer meldet sich betriebsbereit mit READY.

Dies ist kein Wunder, sondern die Eigenschaft des seriellen Bus Ihres Computers. Sie haben vorhin gelernt, wie man den Bus des Computers in Maschinensprache bedient. Dabei fielen auch Worte wie TALK, LISTEN, SENDEN und EMPFANGEN. Der Trick des Spooling ist nun der: Mit Hilfe des CMD-Befehls in Basic können Sie ein Listing auf Diskette »umleiten«, und zwar geschieht dies ähnlich wie beim Drucker: Sie eröffnen ein File und schicken mit dem CMD-Kommando alle weiteren Bildschirmausgaben auf den Bus. Nur ist jetzt nicht der Drucker der Adressat sondern die Floppy. Hier ein Beispiel:
Sie haben ein Listing im Speicher und wollen dieses auf Diskette ablegen, sein Name soll »TEST« sein:
OPEN 1,8,2,"TEST,U,W"
CMD1
LIST

Drucken ohne Umwege

Nach dieser Befehlsfolge wird Ihr Listing als USR-File auf Diskette geschrieben. Wie wäre es nun, wenn die Floppy ein TALK-Kommando erhalten würde, das sie veranlaßt, das eben geschriebene File auf den Bus zu bringen? Der »Hörer« ist aber jetzt nicht, wie üblich, der Computer sondern der Drucker, den wir zuvor mit einem LISTEN dazu aufgefordert haben. Die Folge wäre das, was Sie sich jetzt schon denken können:

Die Floppy schickt das gesamte Listing über den Bus, und der Drucker, der ja auf Empfang programmiert ist, bekommt dieses Listing und druckt es aus. Der Computer hat mit der ganzen Sache nichts zu tun, da er sich nach Senden der Kommandos »zurückgezogen« hat und bleibt demzufolge frei für weitere Arbeit.

Der Zugriff auf den Bus ist dem Computer natürlich für die Zeit der Übertragung verwehrt, aber Sie können währenddessen intern weiterarbeiten. Ist die Übertragung beendet, so sind beide Peripheriegeräte noch auf Sendung und müssen erst »zur Ruhe gebracht« werden, bevor sie wieder ansprechbar sind. Aber auch das erledigt ein kleines Programm für uns. Sehen Sie sich jetzt einmal Listing 5 an. Es enthält ein Spooling-Programm, das mit
SYS828,"filename"
aufgerufen wird. Danach meldet sich der Computer mit
SPOOLING filename
READY
und der Drucker beginnt zu arbeiten. Ist der Druckvorgang beendet, so tippen Sie noch einmal
SYS828
ohne Filenamen, und die Leuchtdiode an der Floppy erlischt. Es erscheint die Meldung
END OF SPOOLING
READY
Dieses Programm ist, im Gegensatz zu unseren anderen Listings, zum sofortigen Eintippen gedacht.

Wie Sie aus diesem Beispiel sehen, kann es von großem Nutzen sein, wenn Sie das Prinzip des seriellen Bus verstehen und dessen »Verkehrsregeln« kennen, da viele Programme nur deshalb mit geringem Aufwand große Effekte und Nutzen erzielen. Ein weiteres Beispiel in dieser Reihe dürfte wohl HYPRA-LOAD sein, das Sie in Ausgabe 10 des 64'er-Magazins fanden. Dieses Programm nutzt aber noch einige weitere Tricks der Maschinenspracheprogrammierung, die wir in den nächsten Ausgaben besprechen wollen.

Was kommt demnächst

In Teil 4 unseres Kurses wollen wir nämlich in die direkte Programmierung der Floppy einsteigen, das heißt, das Abspeichern von Maschinenprogrammen in ihren Pufferspeicher und das Ausführen derselben. Als Beispiel werden wir unser HYPRA-LOAD ein wenig »zerlegen«, um Ihnen die Möglichkeiten dieser Programmiertechnik nahezubringen.

Bis zum nächsten Mal also noch viel Spaß in der Busprogrammierung und in der Anwendung des Druckerspoolings.

(B. Schneider/K. Schramm/gk)

Comal — eine Einführung (Teil 2)

Nachdem wir in der ersten Folge die Grundkenntnisse erworben haben, um mit Comal umgehen zu können, wollen wir jetzt die ersten kleinen Programme erstellen.

Die Verwandtschaft zwischen Basic und Comal wird uns dabei den Einstieg in die einfache Programmierung sehr erleichtern. Wir brauchen uns also nicht wie beim Erlernen anderer Programmiersprachen mit völlig neuen Befehlsstrukturen herumzuschlagen, sondern kommen weitgehend mit unserem Basic-Wissen aus. Man kann sagen, daß zwischen Basic und Comal eine Art Aufwärtskompatibilität besteht. Basic-Programme lassen sich mit minimalen Änderungen in Comal übersetzen; in der anderen Richtung können allerdings größere Schwierigkeiten auftreten. So sind zwar alle numerischen Funktionen und Operatoren von Basic auch in Comal vorhanden, es gibt jedoch zusätzlich einige Comal-Funktionen, die in Basic nicht vorkommen (Tabelle 1), beispielsweise MOD (Restbildung bei Division). Auch die Datenein- und Ausgabe ist in Comal um einiges komfortabler.

Betrachten wir einmal das folgende kleine Beispiel einer Mehrwertsteuer-Berechnung, zunächst in Basic:

```
10 rem Mehrwertsteuer
20 input "Betrag";b
30 s = b * 0.14
40 b = b + s
50 print "Mehrwertsteuer:";s
60 print "Gesamtbetrag:";b
```

Dies ist zugegebenermaßen ein sehr einfaches Beispiel, und man hätte es auch gut in einer Zeile unterbringen können. Aber sehen wir uns dieses Programm doch einmal in Comal an:

```
10 // Mehrwertsteuer
20 input "Betrag ?" : betrag
30 mwert := betrag * 0.14
40 betrag := betrag + mwert
50 print "Mehrwertsteuer:" mwert
60 print "Gesamtbetrag:" , betrag
```

Man erkennt sofort die sehr große Ähnlichkeit beider Programme. Allerdings gibt es auch einige mehr oder weniger auffällige Unterschiede. Zunächst versteht Comal auch lange Variablennamen, wodurch die Programme generell übersichtlicher werden. Als nächstes fällt die Verwendung von »:=« für die Wertzuweisungen auf. Bei der Eingabe braucht man allerdings nur ein Gleichheitszeichen zu schreiben. Comal merkt dann, was gemeint ist und wandelt das Gleichheitszeichen in »:=« um.

Bei genauerem Hinsehen entdeckt man schließlich noch die Verwendung des Doppelpunktes statt eines Semikolons bei der INPUT-Anweisung und die Verwendung des Kommas statt eines Semikolons bei den PRINT-Befehlen.

Befassen wir uns zunächst mit dem INPUT-Befehl. Genau wie in Basic werden damit Daten vom Benutzer erfragt und an die im Befehl angegebenen Variablen zugewiesen. Mehrere Variablen können dabei durch Komma getrennt eingegeben werden.

Enthält die INPUT-Anweisung nur eine Variablenliste und keinen Text, dann erscheint beim Programmlauf ein Fragezeichen, um dem Benutzer mitzuteilen, daß jetzt eine Eingabe erwartet wird.

Im Unterschied zu Basic kann hinter INPUT nicht nur ein Text in Anführungszeichen stehen, sondern auch ein beliebiger Stringausdruck. Hinter diesem Stringausdruck muß ein Doppelpunkt folgen, und dahinter wiederum die Liste der einzulesenden Variablen. In unserem kleinen Beispiel könnten wir also die Zeile 20 ersetzen durch:

```
20 frage$ := "Betrag"
25 input frage$ : betrag
```

Wir müssen allerdings beachten, daß Strings in Comal dimensioniert werden müssen, da vor dem eigentlichen Programmlauf die Adressen aller Variablen festgelegt werden (siehe Teil 1). Um die Adressen von Stringvariablen aber festlegen zu können, muß Comal deren maximale Länge kennen. Bevor wir also die Stringvariable »frage$« das erste Mal benutzen können, muß eine Dimensionie-

Funktion	Bedeutung	Beispiel	Basic-Äquivalent
ABS	Absolutwert	A := ABS(-5)	ABS
AND	Logisches UND	IF A=3 AND B=4 THEN...	AND
ATN	Arcustangens	X := ATG(PHI)	ATN
CHR$	Zeichen	PRINT CHR$(13)	CHR$
COS	Cosinus	X := COS(ALPHA)	COS
DIV	Integerdivision	X := A DIV B	INT(A/B)
EXP	Exponentialfunktion	E := EXP(1)	EXP
IN	Teilstringsuche	IF A$ IN B$ THEN...	(nicht vorhanden)
INT	Ganzzahliger Anteil	N := INT(1.5)	INT
LEN	Stringlänge	L := LEN(X$)	LEN
LOG	Logarithmus	X := LOG(A)	LOG
MOD	Restfunktion	X := A MOD B	A-INT(A/B)*B
NOT	Logisch NICHT	IF NOT FLAG THEN...	NOT
OR	Logisch ODER	IF FLAG OR A>3 THEN...	OR
ORD	ASCII-Wert	PRINT ORD("A")	ASC
PEEK	Speicher lesen	PRINT PEEK(828)	PEEK
RND	Zufallszahl	N := RND(1)	RND
SGN	Vorzeichen-Funktion	S := SGN(X)	SGN
SIN	Sinus	X := SIN(ALPHA)	SIN
SQR	Quadratwurzel	PRINT SQR(2)	SQR
TAN	Tangens	T := TAN(ALPHA)	TAN

Tabelle 1. Comal-Funktionen und Operatoren

Comal — Teil 2

Anweisung	Bedeutung
CASE <Variable> OF	Beginn einer CASE-Anweisung
WHEN <Wert>	Vergleicht <Variable> mit <Wert>; bei Übereinstimmung wird der darauffolgende Programmabschnitt ausgeführt
OTHERWISE	Der folgende Abschnitt wird ausgeführt, wenn keine WHEN-Anweisung zutraf (optional)
ENDCASE	Abschluß einer CASE-Anweisung
IF <Bedingung> THEN	Beginn einer IF-Anweisung. Bei wahrer <Bedingung> wird der Teil nach THEN ausgeführt
ELIF <Bedingung> THEN	Falls die vorhergehende <Bedingung> nicht zutraf, wird die angegebene neue <Bedingung> getestet (optional)
ELSE	Falls keine vorhergehende <Bedingung> zutraf, wird der Programmteil nach ELSE ausgeführt (optional)
ENDIF	Kennzeichnet das Ende eines IF-Blocks (entfällt beim einzeiligen IF)
FOR <Laufvariable> := <Anfang> TO <Ende> DO	Beginn einer Zählschleife, bei der die <Laufvariable> von <Anfang> bis <Ende> hochgezählt wird.
STEP <Schrittweite>	Angabe einer Schrittweite zum Hochzählen der <Laufvariable> (optional)
ENDFOR	Ende einer FOR-Schleife. Statt ENDFOR kann auch NEXT eingegeben werden.
REPEAT	Beginn einer REPEAT...UNTIL-Schleife
UNTIL <Bedingung>	Bei erfüllter Bedingung wird die Schleife beendet, sonst nochmals durchlaufen.
WHILE <Bedingung>	Der nachfolgende Programmteil wird nur durchlaufen, wenn die <Bedingung> erfüllt ist, sonst wird das Programm hinter END WHILE fortgesetzt.
ENDWHILE	Ende einer WHILE...END WHILE-Schleife, springt stets zurück nach WHILE.
LOOP	Beginn einer Endlosschleife. Der Programmteil zwischen LOOP und ENDLOOP wird ständig wiederholt.
EXIT	Ermöglicht das Verlassen einer LOOP...ENDLOOP-Schleife.
ENDLOOP	Erzeugt stets einen Rücksprung nach LOOP.

Tabelle 2. Kontrollstrukturen in Comal. LOOP...ENDLOOP ist in der Version 0.14 noch nicht implementiert.

rung erfolgen. Dies geschieht, indem wir noch eine weitere Zeile einfügen:

15 dim frage$ of 20

Durch diese Anweisung wird Speicherplatz für eine Stringvariable »frage$« mit einer maximalen Länge von 20 Zeichen reserviert. Die Stringlänge ist in Comal übrigens grundsätzlich nur durch den Speicherplatz begrenzt. Nach »dim text$ of 3000« beispielsweise kann text$ bis zu 3000 Zeichen enthalten.

Doch kommen wir nun zur Print-Anweisung, die im wesentlichen analog zu Basic ist, darüber hinaus aber einige zusätzliche Feinheiten kennt.

Formatierte Ausgabe ohne Probleme

Die einzelnen zu druckenden Argumente (numerische oder Stringausdrücke) werden grundsätzlich durch Komma getrennt. Wünscht man die Ausgabe an einer bestimmten Tabulatorstelle, kann man wie in Basic die TAB(n)-Funktion verwenden. Die durch Komma getrennten Ausdrücke werden normalerweise unmittelbar nebeneinander gedruckt — so, als hätte man in Basic ein Semikolon verwendet. Zum Drucken von Tabellen ist das natürlich nicht besonders sinnvoll. Es ist jedoch mit dem Comal-Befehl »ZONE« möglich, die Spaltenbreite für die Print-Anweisung festzulegen. Mit ZONE 10 erhält man eine Spaltenbreite wie bei Basic.

Zur weiteren Formatierung von Zahlenausgaben kann man »PRINT USING« verwenden. Hinter »USING« muß dabei ein String stehen, der das Ausgabeformat bestimmt. Für jede Ziffernstelle der auszudruckenden Zahl steht in diesem String ein Nummernzeichen »#«. Außerdem kann die Position des Dezimalpunktes angegeben werden. In unserem kleinen Mehrwertsteuer-Programm wäre es zum Beispiel sinnvoll, die Geldbeträge mit zwei Nachkommastellen auszugeben. Dazu ersetzen wir die Zeilen 50 und 60 durch die folgenden vier Comal-Zeilen:

```
50 print »Mehrwertsteuer:«,
55 print using "#####.##":
mwert
60 print "Gesamtbetrag:",
65 print using "#####.##":
betrag
```

Jetzt werden die Beträge rechtsbündig mit fünf Stellen vor und zwei Stellen nach dem Komma (oder besser Dezimalpunkt) ausgegeben. Die beiden zusätzlichen PRINT-Befehle waren nötig, da hinter dem Doppelpunkt im Anschluß an das »USING«-

Statement nur noch numerische Parameter folgen dürfen. Die Konstruktion »PRINT USING"# # #"; "Hallo",5« führt zu einer Fehlermeldung, weil »USING« sich an dem String "Hallo" — etwas salopp gesagt — die Zähne ausbeißt.

Der zur »USING«-Anweisung gehörende Formatierungsstring darf übrigens auch andere Zeichen enthalten. Probieren Sie doch einmal folgende Zeile (im Direktmodus) aus:

PRINT USING "DM # # #.# #": 12.6

Experimentieren Sie ruhig einmal mit diesen Formatierungsmöglichkeiten, auch unter Verwendung des ZONE-Befehls.

Strukturiert programmieren

Jede höhere Programmiersprache kennt sogenannte »Kontrollstrukturen«, um den Programmablauf in Abhängigkeit von bestimmten Bedingungen beeinflussen zu können. In Basic gibt es zwei derartige Strukturen, nämlich die Wiederholung mit FOR...NEXT und die Bedingungsabfrage mit IF...THEN. Die Realisierung der IF-Abfrage in Basic hat dabei zwei entscheidende Nachteile. Zum einen fehlt, zumindest im Commodore-Basic, die Angabe einer Alternative (ELSE-Teil einer IF-Anweisung), zum anderen ist die Beschränkung auf eine Zeile in vielen Fällen sehr störend. Man behilft sich in Basic dann mehr schlecht als recht mit GOTO-Sprüngen vor, nach und innerhalb von IF-Anweisungen, was die Übersichtlichkeit eines Programms nicht gerade fördert.

Comal unterstützt nun strukturiertes Programmieren durch eine Vielzahl von Strukturbefehlen (Tabelle 2). Zur Bildung von Programmschleifen stehen neben der von Basic bekannten FOR...NEXT-Struktur noch WHILE...ENDWHILE und REPEAT...UNTIL zur Verfügung. Am einfachsten davon ist die Schleife mit REPEAT...UNTIL (»Wiederhole ... bis«). Hinter UNTIL muß eine Bedingung stehen. Ist die Bedingung nicht erfüllt, wird die Schleife ab REPEAT wiederholt, und zwar so oft, bis entweder die Bedingung wahr wird, oder bis der entnervte Programmierer die STOP-Taste drückt. Der folgende Vierzeiler wartet zum Beispiel, bis die Taste »X« gedrückt wird.

```
10 DIM EINGABE$ OF 1
20 REPEAT
30 EINGABE$ := KEY$
40 UNTIL EINGABE$ = "X"
```

Die Systemvariable »KEY$« enthält stets die gerade gedrückte Taste. Ist keine Taste gedrückt, ist KEY$ = CHR$(0).

WHILE...ENDWHILE funktioniert ähnlich wie REPEAT...UNTIL, nur steht hier die Bedingung direkt hinter WHILE, wird also überprüft, bevor die Schleife zum ersten Mal durchlaufen wird. Dadurch wird eine WHILE-Schleife möglicherweise nie durchlaufen, nämlich dann, wenn die Bedingung von Anfang an schon nicht erfüllt war. In diesem Fall werden alle Befehle zwischen WHILE und ENDWHILE übersprungen und das Programm nach ENDWHILE normal fortgesetzt.

Das genaue Format der WHILE-Schleife ist »WHILE (Bedingung) DO (Anweisungen) ENDWHILE«.

Mit dem »DO« hat es dabei eine besondere Bewandtnis. Steht hinter dem »DO« in der gleichen Zeile eine Anweisung, dann faßt Comal dies als eine einzeilige WHILE-Schleife auf. In diesem Fall darf kein ENDWHILE mehr folgen, sonst gibt es einen »Fehler in der Programmstruktur«. Mit dieser Kurzform einer WHILE-Schleife und dem Comal-Befehl »NULL« läßt sich sehr elegant eine Warteschleife auf einen Tastendruck aufbauen:

```
10 WHILE KEY$ = CHR$(0) DO NULL
```

Die Anweisung NULL ist eine »Dummy«-Anweisung mit der speziellen Eigenschaft, nichts zu bewirken. Die obige Zeile könnte man also etwas frei übersetzen mit »solange keine Taste gedrückt, tue nichts«.

Neben diesen beiden Schleifenstrukturen gibt es natürlich noch die Zählschleife FOR...TO... ENDFOR, die völlig analog zur FOR...NEXT-Schleife in Basic arbeitet, so daß die Besprechung der Arbeitsweise entbehrlich erscheint.

Entscheidungen fällen

In praktisch jedem Programm müssen logische Entscheidungen, meist sogar in großer Anzahl, getroffen werden. Comal stellt dafür eine sehr mächtige IF...THEN...ELIF...ELSE...ENDIF-Konstruktion zur Verfügung die sich in der Regel über mehrere Zeilen erstreckt und ganze Programmblöcke umfassen kann. Daneben gibt es — wie bei »WHILE« — noch eine einzeilige Kurzform. Diese Kurzform besteht einfach darin, daß hinter dem »THEN« in der gleichen Zeile noch ein Befehl folgt. Das funktioniert dann völlig analog zu Basic, nur mit dem Unterschied, daß in Basic noch weitere Befehle, jeweils durch Doppelpunkt getrennt, in der gleichen Zeile folgen dürfen. Für derartige Fälle — und für Fälle, die man in Basic so gar nicht lösen kann — wird in Comal die mehrteilige Form der IF-Anweisung verwendet.

Bei dieser Form muß die Zeile nach dem »THEN« beendet werden. Dann werden, falls die Bedingung hinter dem IF erfüllt ist, alle folgenden Programmzeilen bis zum Ende der IF-Anweisung ausgeführt. Eine mehrzeilige IF-Anweisung muß immer mit dem Schlüsselwort »ENDIF« beendet werden. Außer »ENDIF« darf die entsprechende Zeile allenfalls noch einen Kommentar (//) enthalten. War die IF-Bedingung nicht erfüllt, dann wird das Programm in der auf das »ENDIF«-Statement folgenden Zeile fortgesetzt.

Doch damit sind wir noch längst nicht am Ende. Die IF-Anweisung kann auch um einen »ELSE«-Teil erweitert werden und hat dann das folgende Format:

IF (Bedingung) THEN (Teil 1) ELSE (Teil 2) ENDIF.

Der Programmteil (Teil 1) wird ausgeführt, falls die (Bedingung) erfüllt war, sonst wird (Teil 2) ausgeführt. Jeder dieser beiden Teile ist ein völlig unabhängiges Programmstück und kann seinerseits auch wieder IF-Abfragen enthalten.

Will man gleich mehrere verschiedene Bedingungen testen, dann kann man das Comal-Schlüsselwort »ELIF« verwenden. »ELIF« ist eine Abkürzung für »ELSE IF« und hat auch die gleiche Wirkung, nur mit dem Unterschied, daß keine zweite IF-Anweisung (zu der dann auch ein zweites ENDIF gehören müßte) eröffnet wird. Das folgende Beispielprogramm testet eine einzugebende Zahl auf bestimmte Werte:

```
10 INPUT "ZAHL ?": ZAHL
20 IF ZAHL = 1 THEN
30 PRINT "EINS"
40 ELIF ZAHL = 2 THEN
50 PRINT "Zwei"
60 ELSE
70 PRINT "WEDER EINS NOCH ZWEI"
80 ENDIF
```

Ich erspare mir — und Ihnen — an dieser Stelle, ein entsprechendes Basic-Programm vorzustellen (GOTO, GOTO, ...).

Für den Fall, daß die zu testenden Bedingungen durch einen Variablenwert dargestellt werden können, ist die CASE-Anweisung vorgesehen. Die Wirkungsweise wird wohl am besten klar, wenn wir unser Beispiel zur IF-Anweisung auf die CASE-Konstruktion umschreiben:

```
10 INPUT "ZAHL ?": ZAHL
20 CASE ZAHL OF
30 WHEN 1
40 PRINT "EINS"
50 WHEN 2
60 PRINT "ZWEI"
70 OTHERWISE
80 PRINT "WEDER EINS NOCH ZWEI"
90 ENDCASE
```

In der Kopfzeile einer CASE-Anweisung wird also eine Variable angegeben, gefolgt vom Schlüsselwort »OF«.

Dann folgen beliebig viele Zeilen mit »WHEN«-Konstruktionen. Hinter WHEN ist immer ein Wert angegeben, der bei der Programmausführung mit dem aktuellen Wert der CASE-Variablen verglichen wird. Wird eine Übereinstimmung festgestellt, dann wird der Programmteil hinter der entsprechenden WHEN-Anweisung bis zum folgenden WHEN ausgeführt. Trifft keine WHEN-Bedingung zu, dann wird der Programmteil hinter OTHERWISE ausgeführt. OTHERWISE ist optional und muß nicht vorhanden sein. Trifft keine WHEN-Bedingung zu und ist kein OTHERWISE vorhanden, dann wird das Programm hinter ENDCASE normal fortgesetzt.

Kommen wir nun noch, sowohl last als auch least, zu einem Befehl, den hartgesottene Spaghetticode-Programmierer schon vermißt haben mögen. Gemeint ist die GOTO-Anweisung, die, obschon weitgehend entbehrlich, auch in Comal noch für Spezialfälle zur Verfügung steht. In Comal wird allerdings nicht zu bestimmten Zeilennummern gesprungen, sondern ein GOTO bezieht sich immer auf ein LABEL. Ein Label ist einfach irgendein Name, wie er auch als Variablenname verwendet werden könnte, gefolgt von einem Doppelpunkt. Vor diesem Namen kann, muß aber nicht, das Schlüsselwort LABEL stehen. Die betreffende Zeile darf nur dieses Label und keine weiteren Befehle enthalten. Wer also von GOTO wirklich nicht loskommt, kann das Warten auf einen Tastendruck auch folgendermaßen programmieren (nicht zur Nachahmung empfohlen):

```
5 DIM A$ OF 1
10 LABEL WARTEN:
20 A$ := KEY$
30 IF A$=CHR$(0) THEN GOTO WARTEN
```

Es sollte auch nicht unerwähnt bleiben, daß man nicht in FOR...ENDFOR-Schleifen, Funktionen und Prozeduren hinein oder aus ihnen hinaus springen sollte. Mit »Funktionen und Prozeduren« ist im übrigen bereits das Stichwort für die nächste Folge gegeben. Bis dahin können Sie sich ja vielleicht die Zeit damit vertreiben, Ihre Basic-Programme in Comal umzuschreiben, und zwar selbstverständlich ohne GOTO! (ev)





Wie die Überschrift schon andeutet, hat sich eine Änderung vollzogen. Das »Ka« für Kassette ist weggefallen. Dafür hat sich »Di« zu Disk gemausert. Die Diskette ist aufgrund ihrer Verbreitung ausgewählt worden. Dafür sind jetzt alle Programme einer Ausgabe (VC 20 und C 64) auf einer Diskette erhältlich.

Eines hat sich aber nicht geändert: der Preis. Die Diskette für eine Ausgabe kostet demnach **29,90** Mark. Sie werden bei einigen Disketten bestimmte Programme vermissen. Deren Autoren konnten sich nicht entschließen, ihr Programm im Rahmen des Leserservice für eine Verbreitung auf Datenträger freizugeben. Bei den Ausgaben 5 und 6 können noch Kassetten (VC ...) bestellt werden. Auf kurze Programme wurde aus Gründen der Übersichtlichkeit verzichtet. Nun noch einige technische Details. Zu den Programmen sind immer die Seitenzahlen angegeben, unter der Sie die Beschreibungen in der entsprechenden Ausgabe finden können. Der Diskette liegen also keinerlei Informationen bei. Lesen Sie daher aufmerksam die Anleitung (ob SYS-Befehle nötig sind, in welcher Reihenfolge geladen werden muß, eventuelle Sprach- oder Speichererweiterungen und ähnliches mehr) in dem jeweiligen Artikel nach. Aus Aktualitätsgründen wird jeweils die abgedruckte Version angeboten. Eventuelle systematische Fehler, die sich noch im Programm befinden können, müssen von Ihnen selbst, nach Studium des Druckfehlerteufelchens, korrigiert werden.

Fehlende Hefte erhalten Sie bei: Markt & Technik Vertrieb 64'er Hans-Pinsel-Str. 2, 8013 Haar

Ausgabe 12/84

Bestell-Nr. CB 022 DM 29,90*

Commodore 64

Synthesizer (AdM)
SMON (2. Teil)
3D-Vier gewinnt
Trace
Stringy
Lader
PET-Simulator
Auto
Listschutz
Hires
Simons Axo (SB)
Kreuzworträtsel

VC 20

Mathematikal Basic (8K>) (LdM)
Fast Tape

Ausgabe 11/84

Bestell-Nr. CB 020 DM 29,90*

Commodore 64

Turtle Grafik (LdM)	S.48
Schachmeister (AdM)	S.50
SMON (1. Teil)	S.59
Floppykurs	S.117
FPLOT-Befehlserweiterung	S.73
Get Koala pic	S.66
Interrupttechnik	S.84
Exsort (UPB)	S.154
Einzeiler	S.158
Simons Basic Befehlserweiterung (SB)	S.90

VC 20

Pseudosprites (8K)	S.76
Laterna Magica (8K)	S.68
Betriebssystem-Erweiterung (24K>)	S.88
Supergrafik (GV)	S.71
VC 20-Kurs (GV>)	S.126

Ausgabe 10/84

Bestell-Nr. CB 019 DM 29,90*

Commodore 64

Finanzmathematik (AdM)	S.68
Hypra-Load (LdM)	S.67
Hardcopy	
Compact 2	S.86
Hardcopy MPS 801	S.82
Hardcopy VC 1526 neu	S.83
Hardcopy Gemini-10X	S.85
Hardcopy FX-80	S.88
Hardcopy VC 1520 farbig	S.84
Apocalypse now	S.106
Supercopy	S.102
Disk-Dump	S.95
Diskettenorganisation	S.97
User-Port-Tastatur	S.92
Maske-(UPB)	S.172

VC 20

Epedemic	S.112
Video-Vorspann	S.81

Ausgabe 9/84

Bestell-Nr. CB 014 DM 29,90*

Commodore 64

Indexsequentielle Adreßdatei, S. 54 — Spring Vogel (LdM), S. 68 — Orgel/Synthesizer (AdM), S. 70 — Sprite Aid +, S. 89 — Screen Change, S. 94 — List-Stop, S. 97 — Renew, Datawandler, S. 102 — Synthetische suchen, S. 104 — Geregelter Zahlungsverkehr, S. 164

VC 20

Schiebung (GV>), S. 77 — Deuzei (8K>), S. 79 — Hardcopy 1520 (GV>), S. 87 — RS232-Interface (GV>), S. 100 — Datawandler (GV>), S. 102

Ausgabe 8/84

Bestell-Nr. CB 013 DM 29,90*

Commodore 64

Castle of Doom, S. 66 — Pac-Boy, S. 89 — Kopplung, S. 73 — User-Port-Display, S. 97 — RS232-Test, S. 77 — View BAM, S. 99 — Görlitz Hardcopy, S. 83 — Milchvieh, S. 156

VC 20

Kudiplo (3K), S. 86 — Print at Restore n (GV), S. 101

Ausgabe 7/84

Bestell-Nr. CB 017 DM 29,90*

Commodore 64

Terminalprogramm, S. 24 — Softwarekatalog, S. 72 — Russvok (SB), S. 76 — Crown No. 1, S. 80 — Space Invaders, S. 81 — 1520 Hardcopy, S. 108 — Centronics Interface, S. 110 — Kurvendiskussion, S. 116 — Copy Rel. Files, S. 132 — Autostart, S. 138 — Strubs (OP u. QP), S. 154

VC 20

Rätsel, S. 122

Ausgabe 6/84

Commodore 64

Bestell-Nr. CB 018 DM 29,90*

Lehrerkalender, S. 64 — Morsetrainer, S. 72 — Supervoc, S. 69 — Grafische Darst. (SB), S. 82 — Hot Wheels, S. 92

VC 20 Bestellnummer VC 008

Movemaster (8K), S. 78 — Ghost Manor (GV), S. 104, Logic Disass. (3K>), S. 108, Underground (LdM 16K), S. 120 DM 29,90*

Ausgabe 5/84

Commodore 64

Bestell-Nr. CB 016 DM 29,90*

Adreß- & Telefonregister, S. 64 — Fahrsimulator, S. 82 — Schatzsucher (LdM), S. 90

VC 20 Bestellnummer VC 007

Relative Datei (8K), S. 69 — Schmatzer (GV) S. 76 — 3D-Grafik (8K), S. 78 — Rallye (28K), S. 128 DM 29,90*

Bedeutung der Abkürzungen

*LdM = Listing des Monats
*AdM = Anwendung des Monats
*SB = Simons Basic
*GV = Grundversion
*GV> = alle Speicherversionen können verwendet werden (einschließlich GV)
*3K = 3-KByte-Speichererweiterung wird benötigt
*8K> = Speichererweiterung größer als 8 KByte wird benötigt.
*UPB = Unterprogrammbibliothek

Bestellungen richten Sie bitte an:
M&T Buchverlag, Hans-Pinsel-Str. 2, 8013 Haar

*Alle Preise inklusive Mehrwertsteuer. Der Versand erfolgt mit offener Rechnung zuzüglich Porto und Verpackung.

Fortsetzung von Seite 38

To Sprite, Copy Sprite To Hires«. Zeitlupengrafiken und Bewegungsstudien wie etwa der Flug eines Vogels, werden innerhalb weniger Minuten zur Realität. Wesentlich unterstützt wird der Programmierer bei seinen Entwürfen durch wichtige Befehle wie »Dot, Line, Box, Circle« und andere bekannte Grafikhilfen.

Weil die Programmierer bei HES-ware mit Vorliebe die Interrupttechnik verwendet haben, lassen sich Text und Grafik beliebig mischen. Zusammen mit der umfangreichen Farbgebung sind so Bilder auf der Diskette speicherbar und stehen bei Bedarf, beispielsweise in einem Adventure, rasch zur Verfügung.

Fast ebenso umfangreich wie der Befehlssatz für die Sprite-Programmierung sind die neuen Kommandos zur Tonerzeugung. Alle Musikfunktionen werden wie die Sprites interruptgesteuert. Man kann sich sogar beim Programmieren mit Musik unterhalten lassen. Alle wichtigen Parameter wie Attack, Sustain, Release, Decay und Wellenform werden über simple Basic-Befehle eingestellt.

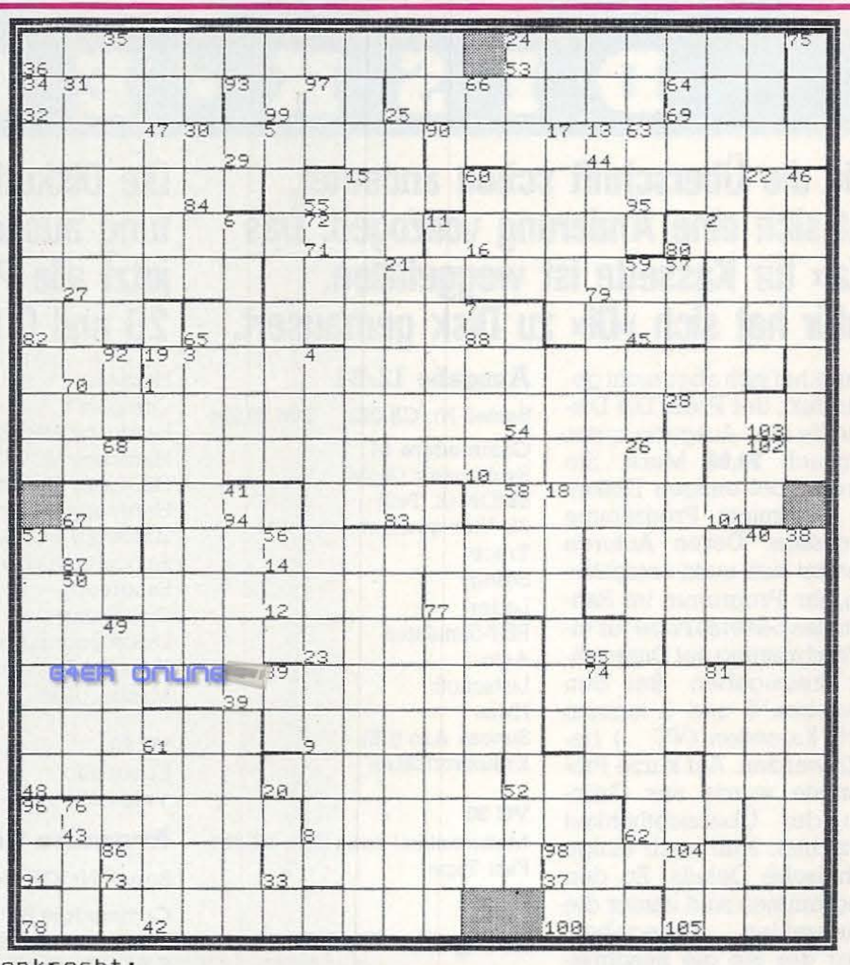
Nach all diesen Erläuterungen entsteht vielleicht der Eindruck, bei »Graphic-Basic« handle es sich um eine reine Grafikerweiterung. Weit gefehlt!. Es gibt Befehle zur Funktionstastenbelegung, ebenso wie solche zur Abfrage der Joysticks und zur Diskettenhandhabung (Dir, Disk). Auch über die typischen Befehle vieler Erweiterungen für bedingte Sprünge (If-Then-Else oder On-Error-Goto) verfügt die Hes-Grafik. Am ungewöhnlichsten ist aber der »Window«-Befehl, der es erlaubt, den Bildschirm in mehrere »Aktivitätszonen« zu unterteilen. Abschließend soll noch angemerkt werden, daß sowohl Text als auch Grafik auf verschiedenen Druckern (Copy) ausgegeben werden kann.

Wer die Grafik- und Tonprogrammierung zu seinem Lieblingsthema gewählt hat, aber auf den Komfort einiger Hilfen zur strukturierten Programmierung nicht verzichten möchte, findet in der Hes-Grafik den richtigen Partner. Für relativ wenig Geld erhält er ein Werkzeug, mit dem der eigenen Kreativität keine Grenzen mehr gesetzt sind. Für meinen C 64 ist »Graphic-Basic« jedenfalls zum gern gesehenen Prinzen geworden.

(Arnd Wängler/Martin Gaksch/ev)

KREUZWORTRÄTSEL SELBER MACHEN –

Kreuzworträtsel sind bei jung und alt beliebt. Das beweisen die vielen Rätselzeitschriften. Aber ein Kreuzworträtselprogramm zu schreiben ist etwas ganz anderes, als ein Rätsel zu lösen.



Senkrecht:

(2) Im Dialog mit d. Computer arbeiten; (3) Speicherzelle mit direktem Zugriff; (4) Folge von Buchstaben/Ziffern; (5) Programmdarstellung auf Papier; (6) Einheit des elektrischen Stroms; (7) Engl. f. 'Stapel'; (11) Tastenfolge b. engl. Tastaturen; (13) Menge der verwendbaren Symbole; (15) Sammelbegriff f. Computerbauteile; (17) Rueckkehr in den Anfangszustand; (18) Spannungs- oder Stromsignal; (19) Billiger Bandspeicher; (21) Jargon f. 'Diskettenstation'; (22) Ein Punkt auf dem Grafikbildschirm; (24) Hoehere strukturierte Progr.Sprache; (26) Vom Compiler erzeugter Kode; (28) Folge v. Anweisungen an d. Computer; (30) Basic-Befehl z. Laden des Speichers; (31) Variable z. Uebergabe an Unterprgr.; (34) Simulation eines fremden Computers; (35) Druckerteil f. Endlospapier; (38) Medium zum Festhalten von Daten; (40) Symbolische Einheit f.Wertzuweisung; (41) Parallele Drucker-Schnittstelle; (46) Logik mit diskreten Zustaenden; (47) Billiger Massenspeicher; (49) Logische Informationseinheit; (50) Symbolischer Name f. eine Adresse; (51) Einzelne logische Funktion; (56) Populaeres Betriebssytem f. Z80-uP; (57) Abk. f. 'Digital nach Analog'; (58) Verbindungsleitung im Prozessor; (59) Abk. f. 'Analog nach Digital'; (60) Hersteller von Mikrocomputern; (63) Steuerzeichen f. Kontrollcodes; (64) Mikroschalter auf der Platine; (66) ASCII-Zeichen f. Zeilenvorschub; (68) Verzweigungsanweisung in BASIC; (72) Strahlungseinheit; (74) Abk.f. 'hoeherwertiges Bit'; (75) Abk.f.'Exklusiv-Oder'-Verknuepfung; (76) Abk.f. 'Megahertz'; (81) Engl. f. Maedchen; (86) Neues Testament; (89) Evangelisch; (90) Grand Prix; (92) Volkswagen; (93) Deutscher Fussballbund; (96) Kurzbez. f. 'lichtjahr'; (97) Doktor; (98) Kurzbez. f. 'Technische Univers.'; (102) Firma; (104) Interessengemeinschaft

Waagerecht:

(1) Aufgabe dieses Wettbewerbs; (8) Wichtiges Eingabegeraet; (9) Programm zur Texteingabe/ Korrektur; (10) Schreibmarke auf dem Bildschirm; (12) Basic-Befehl zum Lesen d. Speichers; (14) Pruefsumme einer Bytefolge; (16) Leicht lernbare Progr.-Sprache; (20) Meldung ueber einen Geraetzustand; (23) Die haelfte eines Bytes oder 4 Bit; (25) Programm zum Starten von Computern; (27) Geraet z. Computerverb. per Telefon; (29) Einheit f. d. elektr. Kapazitaet; (32) Wiederverwendbares ROM; (33) Bildschirmausschnitt; (36) Datentransfer m. mehreren Leitungen; (37) Programm zur Steuerung v. Geraeten; (39) Sich selbst aufrufendes Unterprogr.; (42) Jargon fuer 'Computer'; (43) Punktfeld f. Zeichendarstellung; (44) Adresse die andere Adr. verweist; (45) Logische Informationseinheit; (48) Befehl zur Beendigung v. Unterprgr.; (52) Hoehere kommerzielle Prgr.Sprache; (53) Mech. Geraet z. Zeichnen v. Bildern; (54) Logische Speichereinheit; (55) Hochintegrierter Baustein; (62) Abk. f. 'Zeichen loeschen'; (65) Abk.f. 'Zeilendrucker'; (67) Technik zur Herstellung von IC's; (69) Abk. f.'Disketten-Betriebssystem'; (70) Abk. f. 'Television'; (71) Les- und schreibbarer Speicher; (73) Abk. f. 'Nanosekunde'; (77) Progr.Sprache f. Listenverarbeitung; (78) Abk. f. 'Hertz'; (79) Abk.f. Computer-gestuetzter Entwurf; (80) Griechischer Buchstabe; (82) Dritter Ton vom Grundton; (83) Abk. f. ein engl. Laengenmass; (84) Abk. f. 'Oberbuergermeister'; (85) In Ordnung; (87) Abk. f. 'Sankt'; (88) Seemeile; (91) Abk. f. 'Jahrhundert'; (94) Corps Consulaire; (95) Sozialdemokrat. Partei Deutschl.; (99) Nordeutscher Rundfunk; (100) Kurzbez. f. 'Universitaet'; (101) Kilo-Ampere; (103) Abk. f. 'Hektar'; (105) Abk. f. 'gegruendet'

Bild 1. Das Ergebnis eines Programmlaufs. Während das Programm läuft, werden die vom Computer eingesetzten Wörter direkt in das Rätsel am Bildschirm sichtbar eingetragen. Das geht so schnell, daß Sie mit dem Auge kaum nachkommen können. Erst im Ausdruck werden alle Begriffe durch Zahlen ersetzt. Die Lösung dieses Rätsels finden Sie übrigens auch in diesem Heft an anderer Stelle.

1000 MARK FÜR DEN GEWINNER

Diese Aufgabe ist eine Herausforderung für gute Programmierer. Das beste Ergebnis des Kreuzworträtsel-Wettbewerbs stellen wir Ihnen vor.

Es läßt kaum Wünsche offen. Der Bildschirmaufbau ist genauso gut gelöst wie die Druckerausgabe. Auch den Bedienungskomfort kann man sich kaum besser vorstellen.

Ein Kreuzworträtselprogramm zu schreiben ist eine knifflige Aufgabe. Nicht nur die für den Betrachter eines Rätsels selbstverständlichen Regeln, wie das Kreuzen von Worten senkrecht zueinander und die dichte Vernetzung von Wörtern, auch die Geschwindigkeit und der Komfort des Programmes spielen eine große Rolle. Damit blieb in diesem Falle nur der Einsatz von Maschinensprache übrig. Das Kreuzworträtselprogramm besteht also aus zwei Teilen. Einem Rahmenprogramm in Basic, sowie einer Maschinenroutine, die im Bereich ab $C000/49152 geladen wird und die die Schwerarbeit leistet (zirka 30 Wortvergleiche/Sekunde).

Um das Rätselprogramm nicht unnötig mit der Eingabe des Wortschatzes zu belasten, werden die Wörter vorher mit einem Editor eingegeben und bis zu 255 in einem File auf Diskette abgelegt. Das eigentliche Rätselprogramm liest diese Files dann ein und verwendet sie im Rätselfeld (Bild 1). Trotzdem ist die Eingabe von Hand möglich. Denn der große Wortschatz eines Menschen kann niemals auf Diskette gespeichert werden.

Die Programme

Bevor also die Erzeugung eines Kreuzworträtsels starten kann, muß ein Wortschatz mit dem Lexikon-Editor eingegeben werden (Bild 2). Dieses Programm hat vier Aufgaben:

a) Directory lesen. Dabei werden nur die bereits auf Diskette vorhandenen Wortschatzdateien angezeigt.

b) Wortdatei anlegen. Mit diesem Programmteil können Sie bis zu 255 Wörter und die zugehörigen Fragestellungen eingeben und diese dann auf Diskette als Wortdatei ablegen. Eine spezielle Eingaberoutine (Zeile 1000 — 1099) sorgt dafür, daß nur erlaubte Zeichen eingegeben werden. Es ist zum Beispiel unsinnig, ein Kreuzwort einzugeben, das aus mehreren Teilen besteht, wie etwa »Level II Basic«, oder eine Mischung aus Zahlen und Buchstaben, wie »C 64«.

So können Sie sich also im Laufe der Zeit eine ganze Bibliothek von Wortschätzen aufbauen, die beliebig während der Rätselerzeugung einzusetzen sind.

Eine Besonderheit ist die Verkettung von Dateien. Sie können Nachfolgedateien benennen, die im Rätselprogramm automatisch nachgeladen werden, sobald der Vorgängerwortschatz erschöpft ist. Beispiel: Sie wollen ein Rätsel zum Thema »Computer« basteln. Geben Sie jetzt mit Hilfe des Lexikon-Editors den ersten Teil des geplanten Wortschatzes ein und benennen ihn mit »Computer.1«. Der Folgewortschatz soll »Computer.2« sein. Nachdem Sie den 1. Teil abgespeichert haben, geben Sie den 2. Teil ein und benennen den Nachfolger mit »Computer.3« und so weiter. Wollen Sie diese Kette beenden, wird einfach ein »q« (Quit) für den Nachfolger angegeben. Das Rätselprogramm bricht dann später an dieser Stelle mit dem automatischen Nachladen ab.

Bei dieser Methode hängt die Qualität des Rätsels entscheidend von der Staffelung der Dateien ab, das heißt im ersten File sollten lange Wörter stehen und mit jedem Folgefile die durchschnittliche Wortlänge abnehmen. Die Erstellung des Rätsels gerät dadurch wesentlich flüssiger, denn mit zunehmender Dichte des Wortfeldes müssen die Wörter natürlich immer kürzer werden, damit sie noch einzubauen sind.

c) Wortdatei ändern. Mit Sicherheit tauchen irgendwann falsch eingegebene Kreuzwörter auf, so daß eine Datei verbessert werden muß. Unter diesem Programmpunkt kann daher die entsprechende Wortdatei geladen und editiert werden.

d) Abbrechen. Mit Verlassen des Editors kann das Rätselprogramm geladen und gestartet werden, so daß ein fließender Übergang von der Worteingabe zur Rätselerstellung möglich ist.

Nun zum Kreuzworträtselprogramm. Wie schon gesagt, besteht dieses Programm aus zwei Teilen, einer Maschinenroutine und dem Basic-Rahmenprogramm. Dieses Rahmenprogramm (Bild 3) will ich zuerst beschreiben.

Nach dem Start des Programmes beginnt zunächst eine Initialisierungsphase (Zeilen 27—39). In einem Unterprogramm (ab Zeile 970) werden die Maschinenroutine für das Kreuzworträtsel und Treibersoftware für den Drucker am User-Port geladen. Wenn Sie einen Drukker am IEC-Bus betreiben, ist dieser Programmteil (Zeile 976+979) natürlich nicht nötig und zu entfernen.

Im folgenden Unterprogramm zur Dimensionierung der Variablen und Definition einiger Konstanten (Zeile 915—969) erfolgt der erste Sprung in die Maschinenroutine (Zeile 937), um sie zu initialisieren.

Nach dem Bildaufbau (Unterprogramm ab Zeile 600), ist auf dem Monitor ein Kreuzwortfeld von 20x20 Zeichen zu sehen. Das Programm fragt jetzt nach dem Startwort, das als erstes Wort im Feld eingetragen werden muß, um der Maschinenroutine einen Kristallisationspunkt zu bieten (Zeile 34—35). Geben Sie ein möglichst langes Wort ein, das Sie zum Beispiel unbedingt im Rätsel verarbeiten möchten. Bei dieser Eingabe findet, wie schon beim Editor, ein Zei-

Fortsetzung auf Seite 158

Zur Person des Autors:

Gert Büttgenbach, geb. 20. 09. 53, Beruf: Nautischer Offizier.

Wer sich für eine Berufsausbildung als Deckoffizier in der Handelsschiffahrt entscheidet, so wie ich 1977, der ist ein potentielles Opfer der Computersucht. Denn auf der Brücke eines Frachters kann man eine ganze Menge schon mit einem Taschenrechner zaubern. Die Navigation bietet da ein weites Feld. So dauerte es auch kein halbes Jahr und ich stand klopfenden Herzens vor der schillernden Vitrine eines Kaufhauses in Houston/Texas und vernarrte mich in eines der ersten Exemplare des TI59. Ich mußte mir eine ganze Monatsheuer bei meinem Kumpel pumpen, um in den 7. Bytehimmel aufzusteigen. Von nun an gab es keine Langeweile mehr. Bald berechnete mein Rechenknecht simultan aus 7 Sternenbeobachtungen den Standort auf See.

Während meines Seefahrtstudiums an der Fachhochschule Hamburg lernte ich dann andere Computerfreaks kennen. Inzwischen marmelte ein TRS80 Modell I in meiner Bude und ich mußte erst einmal lernen, daß ein Computer nicht alles kann. Dabei ging der Respekt vor diesen Dingern endgültig flöten. In die weite Welt der CBMs bin ich durch meinen Physikprofessor geraten. Gemeinsam mit anderen Studenten bildeten wir bald einen harten Kern, der der zweitschönsten Sache auf der Welt ungehemmt frönte. Als das Ende des Studiums nahte, war das Thema meiner Abschlußarbeit Schuld an der »Ehe« mit dem C 64. Wir brauchten ein Grafikwunder für die Simulation eines Radarbildes. Und da kam gerade ein merkwürdiger »Spielcomputer« (man konnte einen Joystick anschließen, sehr verdächtig!) über den Speicherhorizont, der mehr zu können schien ...

Wie kam ich nun auf die verrückte Idee, ein Kreuzworträtsel-Programm zu schreiben?

Das hatte ich schon auf Modell I probiert, und gemerkt wie vielfältig die Probleme dabei sind. Als ich vom Preisausschreiben im 64'er-Magazin las, war es natürlich eine Herausforderung, die alte Nuß zu knacken.

Mein nächster Computer? Sorry, aber das könnte ein Macintosh sein, denn mein Buckel wird langsam krumm vom Maschinendenken!

(Gert Büttgenbach/gk)





```
10 print"[clr]";:t=ti+180
11 print"┌──────────────────────────┐
12 print"│ lexikon-editor  hh,30.5.84 │
18 print"└──────────────────────────┘
19 ifti<tthen19
20 :
22 :
24 rem" hauptprogramm
26 rem" ─────────────
28 :
30 gosub900:rem" vorbereitungen
32 :
34 gosub600:rem" menue
36 :
38 m=0
40 iff$="▒"then    gosub3006:goto34:rem
f1
42 iff$="▒"then    gosub0300:goto34:rem
f3
44 iff$="▒"thenclr:gosub0500:goto34:rem
f5
46 iff$="▒"thenprint"[clr]":rem f7
47 :
48 print " Sicher           (j/n) ?"
50 wait203,64:poke198,0:wait203,63
52 getf$:iff$<>"j"then34
53 :
54 printchr$(9);
60 print"[crsd] Raetsel-Programm (j/n) ?
"
62 wait203,64:poke198,0:wait203,63
63 :
64 getf$:iff$="n"then90
66 iff$<>"j"then62
68 print"[clr]load"+chr$(34);
70 print"raetsel";chr$(34);",8"
72 print"[crsd* 3 ]":print"run"
74 poke198,3:poke631,19
76 poke632,13:poke633,13
77 goto97
88 :
90 poke792,71:rem" restore und
92 poke788,49:rem" stop reparieren
94 :
97 end
98 :
99 :
300 rem" datei aendern
302 rem" ─────────────
304 :
306 gosub700:rem" Datei einlesen
307 :
308 ef=0:print"[clr]"
309 print" Mit [rvon]RETURN[rvof] auswae
hlen"
310 print" Mit [rvon]LEERTASTE[rvof]  we
iter"
311 print" Mit [rvon]q[rvof]         abbre
chen"
312 gosub2000:rem" Datei auflisten
313 :
314 print"[crsd]   [rvon]Fertig ?[rvof]
(j/n)"
316 wait203,64:poke198,0:wait203,63
318 getf$:iff$="n"then308
320 iff$<>"j"then316
322 :
323 ifef=0then397
324 gosub800:rem" Datei aufzeichnen
395 :
397 return
398 :
399 :
500 rem" datei eingeben
502 rem" ──────────────
503 :
504 print"[clr]";:gosub900
505 :
506 t$="[rvon]Wortschatz-Name:"
507 l=14:m=1:f$="":gosub1000
508 af$="+"+chr$(20)+f$
510 :
519 print
520 t$="[rvon]Folge-Wortschatz (q=keinen
):"
521 l=14:m=1:f$="":gosub1000
522 nf$="+"+chr$(20)+f$
524 :
525 print"[clr]":nr=0:ef=0
527 :
528 rem" woerter eingeben
529 gosub650
530 :
532 iff$="q"thennr=nr-1:goto539
534 ifnr<254thennr=nr+1
536 ifpeek(210)=7thenprint"[clr]"
537 goto529
538 :
539 ifef=0then597
540 rem" datei auf diskette ablegen
541 fg=nr:gosub800
592 :
597 return
598 :
599 :
600 rem" menue
602 rem"
604 :
606 print"[clr]";:f$=""
608 print"[crsd] [rvon]F1[rvof]  Directo
ry
610 print"[crsd] [rvon]F3[rvof]  Datei a
endern
612 print"[crsd] [rvon]F5[rvof]  Datei a
nlegen
614 print"[crsd] [rvon]F7[rvof]  Abbrech
en
616 :
618 wait203,63:getf$
620 iff$<chr$(133)orf$>chr$(136)then618
621 print"[clr]"
622 :
647 return
648 :
649 :
650 rem" kreuzwoerter eingeben
651 rem" ─────────────────────
652 :
653 n$=str$(nr):n$=right$(n$,len(n$)-1)
654 t$="[rvon]"+n$+". Kreuzwort[rvof] (q
=ende)"
655 l=20:m=0:f$=kw$(nr):gosub1000
656 iff$="q"then697
657 iflen(f$)<2thenprint"[crsu* 3 ]";:go
to655
658 kw$(nr)=f$:print
659 :
664 t$="[rvon]Fragestellung[rvof]"
666 l=35:m=1:f$=fw$(nr):gosub1000
681 fw$(nr)=f$:print
682 :
683 ef=1
686 :
697 return
698 :
699 :
700 rem" datei einlesen
702 rem" ──────────────
703 :
705 print"[clr]"
706 t$="[rvon]Wortschatz[rvof] ($=direct
ory)"
708 l=14:m=1:f$="":gosub1000
709 iff$="$"thenm=1:gosub3000
710 iff$=""then705
712 :
713 af$="+"+chr$(20)+f$
714 open1,8,2,"0:"+af$+",s,r"
716 :
718 input#1,fg:rem" feldgroesse
719 :
720 rem" woerter/fragen einlesen
721 forn=0tofg
722 :input#1,kw$(n),fw$(n)
724 next
```

```
15 print"[clr]";
16 print"┌────────────────────────┐
17 print"│kreuzwortraetsel version 1.a│
18 print"│                          │
19 print"│    von buettgenbach    1984│
20 print"└────────────────────────┘
22 :
23 :
24 rem" ******** hauptprogramm ********
25 rem" *******************************
26 :
27 rem" ────── vorbereitungen ───────
28 clr
29 gosub970 :rem" routinen laden
30 gosub915 :rem" vorbereitungen
32 gosub600 :rem" bild aufbauen
33 :
34 t$="Startwort" :rem" 1.wort
35 nr=1:gosub654  :rem" eintragen
36 :
37 gosub700 :rem" wortschatz einlesen
38 poke53280,0:poke53281,0
39 :
40 rem" ******** hauptschleife ********
42 :
44 poke142,bs:poke26,0:u=usr(dl)
46 :
47 on u goto 50,64,70,80
48 :
49 rem" ────── taste war gedrueckt ─────
50 getf$
51 iff$="▒"thengosub650:goto59
52 iff$="▒"thengosub740:goto59
53 iff$="▒"thengosub700:goto60
54 iff$="▒"thengosub450:goto60
55 iff$="▒"thengosub092:goto59
56 iff$="▒"thengosub087:goto59
57 iffg>0then44
58 goto50
59 print"[home]";ll$;ll$
60 poke53280,0:poke53281,0:gosub337
61 iffg>0then44
62 goto50
63 :
64 rem" ──── suche war vergeblich ─────
65 ifbs=0thengosub110
66 bs=0:goto44
69 :
70 rem" ── ein wort wurde gefunden ───
71 w=peek(25)
72 fs$(nr)=str$((peek(140)=0))+fw$(w)
73 gosub300:nr=nr+1:iffg>0then44
75 :
76 print"[home]";ll$;"[home]";tab(17);
77 print"[cyan][rvon]Wortschatz erschoep
ft !"
78 gosub337:goto50
79 :
80 rem" ──── max. ordn.zahl erreicht ───
81 print"[home]";ll$;"[home][cyan][rvon]
Raetsel fertig:[rvof]";
82 print"  Ausdrucken mit [rvon]RETURN[r
vof]"
83 wait203,64:wait203,63
84 print"[home]";ll$;ll$:gosub740
85 goto95
86 :
87 rem" ────── raetsel loeschen ───────
88 f$="Raetsel loeschen":gosub4000
89 iff$<>"j"thenreturn
90 run30
91 :
92 rem" ───── programm abbrechen ──────
93 f$="Abbrechen":gosub4000
94 iff$<>"j"thenreturn
95 gosub500:rem" programm-ende
96 :
97 end
98 :
99 :
100 rem" ******* unterprogramme *******
102 rem" *****************************
104 :
106 :
110 rem" autom. nachladen
112 rem" ──────────────────────────────
114 :
115 print"[home][cyan]";ll$;"[home]";tab
(18);
116 print"[rvon]Wortschatz ungenuegend"
117 gosub337
118 ifright$(nf$,1)="q"then154
120 :
122 dl=int(dm/3)
124 :
125 ifav=0thenav=1:goto154
126 ifls=0then154
127 :
128 rem" ─── folge-wortschatz laden ───
130 poke53280,2:poke53281,2
132 print"[home]";ll$;"[home][cyan]Worts
chatz ";
133 print"[rvon]";nf$;"[rvof] wird gelad
en"
136 f$=nf$:gosub713
137 poke53280,0:poke53281,0
138 :
154 return
156 :
158 :
300 rem" wortliste kuerzen
302 rem" ──────────────────────────────
304 :
307 sa=peek(47)+peek(48)*256+7
308 :
315 gosub350 :rem" woerter kuerzen
316 :
320 sa=sa+772
322 gosub350 :rem" fragen kuerzen
324 :
334 fg=fg-1:poke2,fg:rem" feld kuerzen
336 :
337 print"[home][whit]";nr;int(fg/og*100
);"[crsr] %  "
338 :
341 return
```

```
726 :
728 input#1,nf$:rem" folge-wortschatz
730 :
732 close1
796 :
797 return
798 :
799 :
800 rem" datei aufzeichnen
802 rem" ────────────────
803 :
805 print"[clr]"
807 :
812 open15,8,15,"i":close15
813 :
814 open1,8,2,"@0:"+af$+",s,w"
816 :
818 print#1,fg :rem" feldgroesse
819 :
820 forn=0tofg
822 :print#1,kw$(n):print#1,fw$(n)
824 next
826 :
828 print#1,nf$:rem" naechster file
830 :
832 close1
896 :
897 return
898 :
899 :
900 rem" vorbereitungen
902 rem" ──────────────
904 :
906 poke53280,6:rem" rahmen blau
908 poke53281,14:rem" grund hellblau
909 printchr$(14);chr$(8);chr$(144)
910 :
912 poke792,193:rem" restore und
913 poke788,52:rem" stop verriegeln
914 :
915 dimkw$(254):rem" kreuzwoerter
916 dimfw$(254):rem" fragen dazu
986 :
987 return
988 :
989 :
990 :
1000 rem" inputform routine
1001 rem" ────────────────
1002 :
1003 x$=chr$(13):y$=chr$(20)
1004 z$=chr$(34):poke198,0
1005 :
1006 print" ";t$:print"[crsd* 2 ][crsu]
? ";f$;" ";
1007 x=1:iff$<>""thenx=len(f$)+1
1008 :
1009 f$="":forn=xtol+1
1010 :print"[crsr]";f$;"_";
1011 :wait198,1:getf$:poke198,0
1012 :iff$=x$then1025
1013 :iff$=y$andn>1thenn=n-1:goto1010
1014 :ifn>1then1011
1015 :ifm=0then1021
1016 :iff$=","orf$=z$then1011
1017 :iff$>=" "andf$<="9"then1022
1018 :iff$>="A"andf$<="Z"then1022
1021 :iff$<"a"orf$>"z"then1011
1022 next
1023 :
1025 printchr$(20);:poke211,1
1026 poke631,13:poke198,1:inputf$
1030 iff$=x$thenprint"[crsu][crsl* 4 ]";
:goto1009
1036 :
1097 return
1098 :
1099 :
2000 rem" datei auflisten
2002 rem" ───────────────
2004 :
2010 fornr=0tofg
2012 :print"[crsd] ->[rvon]";kw$(nr);"[r
vof]"
2014 :print"   ";fw$(nr)
2016 :forw=0to60:next
2017 :wait198,1:getf$:print"[crsu* 2 ]
 [crsd* 2 ]"
2018 :iff$="q"then2029
2019 :iff$<>chr$(13)then2024
2021 :print"[clr]":gosub650
2022 :print"[clr]":wait203,64
2024 :ifpeek(210)=7thenprint"[clr]"
2026 next
2028 :
2029 t$="[rvon]Folge-Wortschatz"
2030 print"   ";t$;":":print" ->";nf$
2032 wait203,64:poke198,0:wait203,63
2034 :
2036 getf$:iff$<>chr$(13)then2097
2037 f$=right$(nf$,len(nf$)-2)
2038 t$=t$+" (q=keinen):"
2039 print"[clr]":gosub1000:ef=1
2040 nf$="+"+chr$(20)+f$
2096 :
2097 return
2098 :
2099 :
3000 rem" directory listen
3002 rem" ────────────────
3003 :
3004 print"[clr]"
3005 print" Mit [rvon]RETURN[rvof] auswa
ehlen"
3006 print" Mit [rvon]LEERTASTE[rvof]  w
eiter"
3007 print
3008 :
3009 open15,8,15,"io":open2,8,2,"#"
3010 :
3012 t=18:s=1:f$=""
3014 :
3016 forw=0to99
3018 :
3020 :print#15,"b-r";2;0;t;s
3022 :print#15,"b-p";2;0
3024 :
3026 :get#2,x$:t=asc(x$+chr$(0))
3028 :get#2,x$:s=asc(x$+chr$(0))
3030 :
3032 :forx=0to7
3034 : print#15,"b-p";2;x*32+5
3036 : get#2,x$
3038 : ifx$<>"+"thennextx:goto3060
3040 : get#2,x$
3042 : fory=1to14
3044 :  get#2,x$:f$=f$+x$
3046 :  ifx$=chr$(160)theny=15
3048 : nexty
3050 : print" -> [rvon]";f$;"[rvof]"
3051 : ifm=0then3056
3053 : wait203,64:wait203,63:getx$
3054 : ifx$=chr$(13)then3064
3055 : print"[crsu]    "
3056 : f$=""
3057 :nextx
3058 :
3060 :ift=0thenw=99
3061 nextw
3062 :
3064 close2:close15
3065 ifm=0thenwait203,63
3066 :
3068 return
3070 :
3071 :
3072 :
9000 rem"┌────────────────────────────┐
9002 rem"│ ende       lexikon-editor  │
9004 rem"└────────────────────────────┘

ready.
```


Bild 2. Editor. Mit diesem Programm können Sie Ihre eigenen Wortschatzdateien aufbauen. Sie werden auf Diskette gespeichert.

```
342 :
344 :
349 rem" wortvektoren vertauschen
350 x=sa+fg*3:y=sa+w*3
354 pokey,peek(x)
356 pokey+1,peek(x+1)
358 pokey+2,peek(x+2)
360 :
397 return
398 :
399 :
450 rem" autoladen ein/aus
452 rem" ─────────────────────────
454 :
456 ifls=0thenls=1:goto461
458 ifls=1thenls=0
460 :
461 poke214,13:poke211,30:sys58732
462 :
463 print"[red ]";
464 ifls=1thenprint"[rvon]Ein[rvof]";
466 ifls=0thenprint"Aus";
472 :
497 return
498 :
499 :
500 rem" programm-ende
502 rem" ─────────────────────────
504 :
505 print"[clr]▒";chr$(9);
506 poke53280,14:rem" rahmen hellblau
508 poke53281,06:rem" grund blau
510 :
512 poke792,71:rem" restore und
514 poke788,49:rem" stop moeglich
594 :
596 return
598 :
599 :
600 rem" bild aufbauen
602 rem" ─────────────────────────
604 :
606 poke53265,11 :rem" bild aus
607 :
608 rem" ───── kreuzwortfeld ──────
609 print"[clr][crsd* 3 ]";chr$(14);chr$
(8);
610 print" [cyan]┌──────────────────────┐
"
612 forn=1to20
614 print" [cyan]│[rvon]▒
   [rvof][cyan]│"
616 next
618 print" [cyan]└──────────────────────┘
";
619 :
620 rem" ───── tastenbelegung ──────
621 b$="[crsl* 28 ]"
622 print"[home][crsd* 3 ][red ]";
623 a$="Woerter   einlesen"
624 f=1:gosub639
625 a$="Raetsel   drucken "
626 f=4:gosub639
627 a$="Autolader Aus     "
628 f=6:gosub639
629 a$="von Hand  eingeben"
630 f=7:gosub639
631 printb$;"[crsd][rvon]f 2[rvof][crsd]
crsr* 2 ]Loeschen"
632 printb$;"[crsd][rvon]f 8[rvof][crsd]
crsr* 2 ]Abbrechen";
633 :
634 poke53265,27:rem" bild an
635 :
636 return
637 :
639 printb$;"[rvon]f";f;"[rvof]──────┐
640 rem    printb$;"│         │
641 printb$;"│";left$(a$,9);"│
642 printb$;"│";right$(a$,9);"│"
643 printb$;"└─────────┘
645 :
646 return
648 :
649 :
650 rem" kreuzwort v. hand eingeben
651 rem" ─────────────────────────
652 :
653 t$="Kreuzwort (Abbr=q)"
654 poke53280,2:poke53281,2
655 print"[home]";ll$;ll$
656 l=20:m=0:gosub1000:m$=f$
657 ifm$="q"andnr>1then685
658 l=len(m$):ifl<2then656
659 :
660 t$=m$:gosub2000 :rem" pos. eingeben
661 :
662 poke26,len(m$):m$=m$:rem" wort
663 wait203,64:u=usr(0) :rem" eintragen
664 ifu<>3then685
665 :
666 t$="Fragestellung" :rem" frage
667 l=36:m=1:gosub1000 :rem" eingeben
669 :
680 x$=str$(peek(140)=0)  :rem" frage
681 fs$(nr)=x$+f$:nr=nr+1 :rem" merken
682 av=0:dl=dm
683 goto650
684 :
685 return
698 :
699 :
700 rem" kreuzwoerter einlesen
701 rem" ─────────────────────────
702 :
703 poke53280,2:poke53281,2
704 print"[home]";ll$;ll$;
705 t$="Wortschatz (Abbr=q)"
706 l=14:m=1:gosub1000
707 iff$="q"thenprint"[home]";ll$:goto73
3
708 iff$="$"then710
709 f$="+"+chr$(20)+f$:goto712
710 gosub3000:iff$=""then704
711 :
712 print"[home]";ll$;ll$
```

Bild 3. Das Listing zum Kreuzworträtsel

```
713 open15,8,15
714 open1,8,2,"0:"+f$+",s,r"
715 input#15,en,en$
716 ifen<>0then728
717 :
718 input#1,fg :rem" feldgroesse gleich
719 poke2,fg   :rem" anzahl woerter
720 forn=0tofg :rem"lese woerter/fragen
721 :input#1,kw$(n),fw$(n)
722 next
723 :
724 input#1,nf$:rem" naechster file
725 input#15,en,en$
726 av=0:dl=dm:bs=1:og=fg
727 :
728 print"[home]";left$(ll$,40-len(en$))
;
729 print"[rvon]";en$
730 :
731 close1:close15
732 :
733 return
734 :
735 :
740 rem" raetsel drucken
741 rem" ──────────────────────
742 :
743 poke53280,2:poke53281,2
744 print"[home]";ll$;ll$
745 print"[home][cyan][rvon]Raetsel wird
 ausgedruckt[rvof]"
746 :
747 gosub761:rem" wortfeld ausdrucken
748 gosub824:rem" fragen ausdrucken
749 gosub900:rem" loesung ausdrucken
750 :
751 return
752 :
753 :
754 rem" ── init epson rx-80 drucker ──
755 open1,4:print#1,chr$(27);"@";
756 print#1,chr$(27);"3";chr$(24);
757 rem print#1,chr$(27);"1";chr$(lr);
758 print#1: close1:return
759 :
760 rem" ──── wortfeld ausdrucken ────
761 gosub755:open1,4,10:cmd1 :print:prin
t"[rvon]┌";
762 forn=0to19:print"──";:next
763 print"┐[rvof]"
764 :
765 forze=0to19
766 :d=ze*20:ad=s1+d:as=s2+d:aw=s3+d
767 :ab=s4+ze*40
768 :
772 :print" ";
773 :forsp=0to19
774 : c=peek(as+sp)
775 : ifc=0thenprint"   ";:goto777
776 : printmid$(str$(c)+"   ",2,3);
777 :nextsp
779 :gosub820
780 :forn=1to3
781 : print"[rvon]|[rvof]";
782 : forsp=0to19
783 :  a=166:b=a:c=a
784 :  ifpeek(ab+sp)=160then790
785 :  b=32:c=b:ifsp=19then787
786 :  if(peek(ad+sp)and1)thenc=167
787 :  a=165:ifsp=0thena=32
788 :  if(n=1)andpeek(as+sp)thena=32
789 :  if(n=3)andpeek(aw+sp)thena=32
790 :  printchr$(a);chr$(b);chr$(c);
791 : nextsp
792 : print"[rvon]|[rvof]";:ifn<3thenpri
nt
793 :nextn
794 :
795 :gosub820:print" ";
796 :forsp=0to19
797 : c=peek(aw+sp)
798 : ifc=0thenprint"   ";:goto800
799 : printmid$(str$(c)+"   ",2,3);
800 :next
801 :
802 :ifze=19then810
803 :gosub820:print" ";
804 :forsp=0to19
805 : p=ad+sp:c=(peek(p)and16)
806 : ifcthenprint"───";:goto808
807 : print"___";
808 :nextsp
809 :
810 print:nextze
811 :
812 print"[rvon]└";
813 :forn=0to19:print"──";:next
814 print"┘[rvof]";:print#1:close1
815 :
816 return
817 :
818 :
820 printchr$(141);:return
821 rem  printchr$(141);left$(ll$,lr);:r
eturn :rem fuer rx80
822 :
823 rem" ──── fragen ausdrucken ────
824 printchr$(14):open1,4
825 print#1,chr$(27);"0";
826 print#1,chr$(27);chr$(15);
827 print#1,chr$(27);"l";chr$(lr*1.7);
828 :
829 cmd1:printchr$(14);"Senkrecht:"
830 sw=0:gosub838:print
831 :
832 printchr$(14);"Waagerecht:"
833 sw=-1:gosub838:print#1:close1
834 :
835 return
836 :
837 :
838 mz=0:zl=0:f=0:na=1
839 forn=1tonr-1
840 :ifval(fs$(n))=swthengosub847
841 nextn
842 m=0:mr=0:bz=0:gosub881:print
843 :
844 return
845 :
846 :
847 nl=len(str$(n))+2
848 l=zl+nl+len(fs$(n))
849 ifl<czthenzl=l:goto861
850 :
851 pa=1
852 forp=3tolen(fs$(n))-2
853 :ifmid$(fs$(n),p,1)<>" "then855
854 :l=zl+nl+p-2:ifl<czthenpa=p
855 nextp
856 :
857 ifpa=1thenmz=mz-1:l=zl
858 ifpa>1thenl=zl+nl+pa-2
859 gosub866:mz=0
860 :
861 mz=mz+1
862 :
863 return
864 :
865 :
866 bz=int((cz-1)/mz):mr=cz-(1+bz*mz)
867 m=-(mr>0):gosub881
868 :
869 na=n+1:l=len(fs$(n)):ifpa>1then873
870 w=n:print";":gosub892:zl=nl+l+3
871 printmid$(fs$(n),3,l);:goto878
872 :
873 print"; ";left$(ll$,bz);
874 w=n:gosub892:zl=l-pa+2
875 printmid$(fs$(n),3,pa-2)
876 printright$(fs$(n),l-pa);
877 :
878 return
879 :
880 :
881 forp=naton-1
882 :ifval(fs$(p))<>swthen887
883 :iffthenprint"; ";left$(ll$,bz+m);
884 :iffthenmr=mr+(mr>0):m=-(mr>0)
885 :f=1:w=p:gosub892
886 :printright$(fs$(p),len(fs$(p))-2);
887 nextp
888 :
889 return
890 :
891 :
892 nl=len(str$(w))-1:print"(";
893 printright$(str$(w),nl);") ";
894 return
895 :
899 rem" ──── loesung ausdrucken ────
900 gosub755:printchr$(14):open1,4,10:cm
d1
901 :
902 print:print:print:print"Loesung:"
903 print"┌────────────────────┐"
904 forn=s4tos4+760step40
905 :print"|";:forp=nton+19
906 :printchr$(peek(p));
907 :nextp:print"|"
908 nextn
909 print"└────────────────────┘"
910 print#1:close1
911 :
912 return
913 :
914 :
915 rem" vorbereitungen
916 rem" ──────────────────────
917 :
919 poke53280,2:rem" rahmen grau
920 poke53281,2:rem" hintergrund grau
921 :
922 rem   poke792,193:rem" restore und
923 rem poke788,52 :rem" stop verriegeln
924 :
925 dimkw$(254):rem" kreuzwoerter
926 dimfw$(254):rem" fragen dazu
927 dimfs$(255):rem" fragestellung
928 fg=1:og=fg :rem" feldgroesse
929 poke2,fg   :rem" uebergeben
930 :
931 poke785,0   :rem" usr-vektor
932 poke786,192 :rem" setzen
933 :
934 ls=0:rem" Autolader Aus
935 bs=1:rem" Blausperre Ein
936 :
937 sys50016:rem" matrix loeschen
938 :
940 rem sys 51859 :rem" init. drucktreib
er
942 :
944 rem" konstanten:
945 :
946 lr=10  :rem" linker rand druckausg.
948 cz=int(130-lr*1.7) :rem" zeillaenge
949 :
950 forn=1to40    :rem" leerstring fuer
952 :ll$=ll$+" " :rem" formatierte
954 next          :rem" ausgabe
955 :
956 s1=50176 :rem" adressen der
958 s2=50576 :rem" wortfeld-
960 s3=50976 :rem" speicher
962 s4=1186
963 :
964 dm=10000:rem" max. anz. laeufe
965 :
966 return
967 :
969 :
970 rem" routinen laden
971 rem" ──────────────────────
972 :
975 forn=0to3:s1=s1+peek(49152+n):next
976 forn=0to3:s2=s2+peek(51857+n):next
977 :
978 ifs1<>483thenload"such.obj",8,1
979 rem ifs2<>494thenload"druck.obj",8,1
980 :
987 return
997 :
998 :
999 :
1000 rem" inputform routine
1001 rem" ──────────────────────
1002 :
1003 x$=chr$(13):y$=chr$(20)
1004 z$=chr$(34)
1005 :
1006 print"[home][cyan][rvon]";t$;"?[rvo
f]  ";:poke198,0
1007 :
1008 f$=""
1009 forn=1tol+1
1010 :print"[crsr]";f$;"_";
1011 :wait198,1:getf$:poke198,0
1012 :iff$=x$then1024
1013 :iff$=y$andn>1thenn=n-1:goto1010
1014 :ifn>1then1011
1015 :ifm=0then1021
1016 :iff$=","orf$=z$then1011
1017 :iff$>=" "andf$<="9"then1022
1018 :iff$>="A"andf$<="Z"then1022
1021 :iff$<"a"orf$>"z"then1011
1022 next
1023 :
1024 print"[crsr] [crsr][home][rvon]";t$
;
1026 poke631,13:poke198,1:inputf$
1029 iff$=x$then1005
1030 :
1097 return
1098 :
1099 :
2000 rem" wort positionieren
2001 rem" ──────────────────────
2002 :
2004 print"[home][crsd]▒[rvon] (S) enkre
cht   oder";
2005 print"   (W) aagerecht  ? "
2006 wait203,63:getf$
2007 iff$<>"s"andf$<>"w"then2006
2008 :
2009 print"[home][crsd][rvof]";ll$;
2010 print"[home][crsd][rvon]Bitte Posit
ion anfahren!"
2011 print"[home][crsd* 3 ][crsl* 3 ][rv
of]";
2012 :
2013 fa=1186:p=fa:f=peek(fa)
2014 x=0:y=0:s=(f$="s"):poke140,abs(s)
2015 mx=20+1*(s=0)+(s=-1)
2016 my=20+1*(s=-1)+(s=0)
2017 :
2018 forn=0to9999
2019 pokep,f:p=fa+x+y*40
2020 f=peek(p):pokep,94:wait198,1
2021 getf$:iff$=chr$(13)then2028
2022 iff$="[crsd]"theny=y-1*(y<my):next
2023 iff$="[crsl]"thenx=x-1*(x<mx):next
2024 iff$="[crsu]"theny=y+1*(y>0):next
2025 iff$="[crsr]"thenx=x+1*(x>0):next
2026 next
2027 :
2028 pokep,f
2029 print"[home]";ll$;ll$;
2030 :
```

```
2031 poke211,x :rem" cursor-position
2032 poke214,y :rem" setzen
2033 :
2097 return
2098 :
2099 :
3000 rem" directory listen
3001 rem" ----------------------
3003 :
3004 print"[home]";l1$;"[home][cyan]";
3005 print"Mit [rvon]RETURN[rvof] auswae
hlen "
3006 print"[crsu]Mit [rvon]LEERTASTE[rvo
f] weiter "
3007 :
3008 open15,8,15,"io":open2,8,2,"#"
3010 :
3012 t=18:s=1:f$=""
3014 :
3016 forw=0to99
3018 :
3020 :print#15,"b-r";2;0;t;s
3022 :print#15,"b-p";2;0
3024 :
3026 :get#2,x$:t=asc(x$+chr$(0))
3028 :get#2,x$:s=asc(x$+chr$(0))
3030 :
3032 :forx=0to7
3034 : print#15,"b-p";2;x*32+5:get#2,f$
3038 : iff$<>"+"thennextx:goto3060
3042 : fory=1to15
3044 :  get#2,x$:f$=f$+x$
3046 :  ifx$=chr$(160)theny=15
3048 : nexty
3050 : print"[home]";tab(24);"[rvon]";
3052 : printright$(f$,len(f$)-2);"[rvof]
"
3053 : wait203,64:wait203,63:getx$
3054 : ifx$=chr$(13)then3064
3055 : print"[home]";tab(24);left$(l1$,1
4)
3056 : f$=""
3057 :nextx
3058 :
3060 :ift=0thenw=99
3061 nextw
3062 :
3064 close2:close15
3065 print"[home]";l1$;l1$
3066 :
3068 return
3070 :
3071 :
4000 rem" sicherheitsabfrage
4001 rem" ----------------
4002 :
4003 poke53280,2:poke53281,2
4004 print"[home]";l1$
4006 print"[home][cyan][rvon]";f$;":[rvo
f] Sicher (J/N) ?"
4008 wait203,64:wait203,63:getf$
4010 iff$<>"j"thenprint"[home]";l1$
4012 :
4014 return
4016 :
4018 :
4020 :
9000 rem" ------------------------------
9002 rem"| ende   raetselgenerator     |
9004 rem" ------------------------------

ready.
```

Bild 3. Das Listing zum Kreuzworträtsel. Beachten Sie die Hinweise zum Eintippen. Speichern Sie dieses Programm unter dem Namen »RAETSEL« auf Diskette. Vor dem Starten muß zuerst Listing in Bild 4 eingegeben und gespeichert werden.

```
10 rem" basic-lader fuer kreuzwort-suchr
outine
11 rem" wichtig: vor dem 1.lauf abspeich
ern !
12 :
13 ad=12*4096 :rem" = $c000
14 :
15 for n=0 to 906: read by: poke ad+n,by
: su=su+by: next
16 if su<>105962 then print" checksum -
error" :stop
17 :
20 rem" such-routine abspeichern
22 :
24 poke 43,  0 :rem" vektor auf beginn
26 poke 44,192 :rem" der routine
28 :
30 poke 45,138 :rem" vektor auf ende
32 poke 46,195 :rem" der routine
34 :
36 save"such.obj",8
38 :
40 sys 64738
42 :
44 :
45 data120,32,139,192,240,9,32,188,192,3
2,240,192,76,21,192,32,15,193,32
46 data100,193,176,19,169,0,133,139,141,
0,220,173,1,220,201,255,240,234
47 data160,1,76,135,192,165,140,240,6,32
,21,194,76,55,192,32,95,194,176
48 data45,230,25,165,2,197,25,144,17,162
,1,32,56,193,32,71,193,165,98,197
49 data26,144,234,76,42,192,165,20,208,1
1,165,21,208,5,160,2,76,135,192
50 data198,21,198,20,76,23,192,165,139,2
08,12,165,141,240,228,165,142,240
51 data4,165,143,240,220,238,248,7,32,14
6,194,160,3,173,248,7,201,255,208
52 data2,160,4,88,76,162,179,32,247,183,
165,47,166,48,133,78,134,79,160
53 data0,132,139,169,7,24,101,78,144,2,2
30,79,133,78,169,162,162,4,133,80
54 data134,81,162,216,133,82,134,83,165,
26,201,0,240,2,133,139,96,165,211
55 data133,28,165,214,133,29,24,105,4,13
3,214,32,108,229,165,209,133,87
56 data165,210,133,88,230,87,230,87,32,3
6,234,165,243,133,91,165,244,133
57 data92,230,91,230,91,165,51,[illegible],52,13
3,101,134,102,96,164,28,177,91,41
58 data15,201,14,240,17,170,165,140,208,
7,138,201,1,208,9,240,5,138,201
59 data7,208,2,56,96,24,96,169,0,133,140
,165,2,32,88,193,133,25,165,78,133
60 data99,165,79,133,100,166,25,240,3,32
,56,193,32,71,193,169,255,32,88
61 data193,201,128,144,2,133,140,96,169,
3,24,101,99,144,2,230,100,133,99
62 data202,208,242,96,160,0,177,99,133,2
6,200,177,99,133,101,200,177,99
63 data133,102,96,133,97,173,27,212,197,
97,240,2,176,247,96,162,19,134,31
64 data134,32,232,138,56,229,26,166,140,
240,6,133,32,169,7,208,4,133,31
65 data169,1,133,33,165,31,32,88,193,133
,28,165,32,32,88,193,133,29,165
66 data82,166,83,133,91,134,92,164,29,24
0,14,169,40,24,101,91,144,2,230
67 data92,133,91,136,208,242,164,28,177,
91,41,15,201,14,240,26,197,33,240
68 data22,198,28,16,238,165,31,133,28,16
5,29,197,32,240,6,230,29,160,1,208
69 data208,24,96,165,80,166,81,133,87,13
4,88,164,29,240,14,169,40,24,101
70 data87,144,2,230,88,133,87,136,208,24
2,169,20,166,140,240,6,56,229,29
71 data76,244,193,56,229,28,133,98,56,96
,169,0,133,141,133,143,133,27,165
72 data87,166,88,133,89,134,90,165,91,16
6,92,133,93,134,94,165,28,133,30
73 data96,32,248,193,164,30,177,89,201,1
60,240,8,164,27,209,101,208,53,164
74 data30,177,93,41,15,201,14,208,4,133,
143,240,6,201,7,208,35,133,141,230
75 data27,165,27,197,26,240,27,169,40,24
,101,89,144,2,230,90,133,89,169
76 data40,24,101,93,144,2,230,94,133,93,
24,144,189,24,96,56,96,32,248,193
77 data164,30,177,89,201,160,240,8,164,2
7,209,101,208,32,164,30,177,93,41
78 data15,201,14,208,4,133,143,240,6,201
,1,208,14,133,141,230,30,230,27
79 data165,27,197,26,208,212,56,96,24,96
,32,248,193,164,27,177,101,164,30
80 data145,89,169,5,133,33,177,93,41,15,
201,14,208,12,169,7,133,33,165,140
81 data240,4,169,1,133,33,165,33,145,93,
230,27,165,27,197,26,240,34,165
82 data140,240,25,169,40,24,101,89,144,2
,230,90,133,89,169,40,24,101,93
83 data144,2,230,94,133,93,76,149,194,23
0,30,76,149,194,169,0,162,196,133
84 data87,134,88,164,29,240,14,169,20,24
,101,87,144,2,230,88,133,87,136
85 data208,242,133,89,165,88,133,90,164,
28,177,89,166,140,240,4,9,64,208
86 data2,9,4,145,89,224,0,240,15,24,165,
89,105,144,133,89,165,90,105,1,133
87 data90,208,13,24,165,89,105,32,133,89
,165,90,105,3,133,90,173,248,7,145
88 data89,224,0,240,23,166,26,24,202,240
,13,169,20,101,87,144,3,230,88,24
89 data133,87,144,240,169,16,208,8,152,2
4,101,26,168,136,169,1,17,87,145
90 data87,96,169,0,162,196,133,89,134,90
,160,0,140,248,7,152,145,89,230
91 data89,208,250,230,90,165,90,201,201,
208,241,169,255,141,14,212,141,15
92 data212,169,129,141,18,212,96,0
100 rem-------------------------------
110 rem data tester
120 rem-------------------------------
130 input"blockgroesse";bg
140 gosub240
145 open1,4:cmd1:rem nur bei drucker
150 print"▒anzahl blocksumme gesamtsumme
 text    "
160 gosub240
170 restore
180 reada$:a=val(a$)
190 ifa$="*"thengosub240:printb,s1,s:end
200 an=an+1:s=s+a:b=b+1:s1=s1+a
210 ifa$<>mid$(str$(a),2)thenprint"▒"b,,
,a$"▒"
215 ifa>256thenprint"▒"b,,,a$"▒"
220 ifan=bgthenprintb,s1,s:an=0:s1=0
230 goto180
240 fori=1to10:print"----";:next:return
250 print#1:close1:rem nur bei drucker
340 data*

ready.
```

Bild 4. Dieses Programm erzeugt ein Maschinenprogramm und speichert es unter dem Namen »SUCH.OBJ« auf Diskette. Der Teil ab Zeile 100 ist nicht notwendig. Mit diesem Teil können Sie mit RUN 100 die DATAS überprüfen. Er erzeugte den Ausdruck, wie unten zu sehen ist.

pruefsummenliste
blockgroesse 30

zeile	anzahl	summe
46	30	3459
48	60	7430
49	90	10532
51	120	14928
52	150	18545
54	180	21921
56	210	25654
57	240	29719
59	270	32931
60	300	36070
62	330	39840
63	360	43658
65	390	46780
67	420	49962
68	450	53537
70	480	56765
71	510	60554
73	540	64019
74	570	67584
76	600	70757
78	630	74189
79	660	77740
81	690	80991
82	720	84221
84	750	87844
85	780	91244
87	810	94181
88	840	97365
90	870	100670
92	900	105197
gesamt	907	105962

Fortsetzung von Seite 151

chencheck statt, der es unmöglich machen soll, unerlaubte Zeichen im Rätselfeld unterzubringen. Danach müssen Sie sich zwischen einer waagerechten oder senkrechten Eintragung entscheiden, und ein Cursor taucht im Wortfeld auf. Fahren Sie wie gewohnt mit den Cursortasten die gewünschte Wortposition an und drücken die RETURN-Taste. Da es das erste Wort ist und genügend Platz im Wortfeld herrscht, wird Ihr Wort ohne Protest sofort eingetragen. Jetzt noch schnell die zugehörige Fragestellung eingetippt, und das erste Wort ist korrekt eingetragen.

Sie können dieses Spiel beliebig fortsetzen und theoretisch das ganze Rätsel auf diese Weise per Hand erstellen. Alle Eingaben von Hand sind frei von dem Zwang, ein Wort mit einem anderen kreuzen zu müssen. Sie können Ihre Wörter also beliebig positionieren, sollte ein Wort allerdings nicht passen, wird es zurückgewiesen und der Handeingabe-Modus abgebrochen.

Regulär verlassen Sie die Handeingabe, indem Sie anstelle eines neuen Wortes ein »q« eintippen. Später können Sie die automatische Rätselerzeugung jederzeit unterbrechen und mit F7 wieder in den Handmodus zurückkehren.

Im Wortfeld stehen nun ein oder mehrere Wörter, die als Startpunkte für andere Wörter dienen. Senkrechte Eintragungen sind weiß und waagerechte gelb gefärbt.

Bleibt nur die Angabe, welcher auf Diskette gespeicherte Wortschatz als erstes geladen werden soll (Zeile 37). Dabei können Sie sich mit »$« auch das Inhaltsverzeichnis der Diskette ansehen, für den Fall, daß Ihnen der Name eines Wortschatzes entfallen ist.

Nach erfolgreichem Laden des ersten Wortschatzes beginnt nun die automatische Rätselerzeugung (Hauptschleife Zeile 40—99). Zunächst überraschend schnell füllt sich das Wortfeld mit zufällig plazierten und gekreuzten Wörtern. Dieser Vorgang wird von der Maschinenroutine gesteuert. Auf die genaue Arbeitsweise dieses Programmteiles gehe ich noch gesondert ein. Links oben auf dem Bildschirm erscheint die Anzahl der Wörter, die bereits eingetragen sind (maximal 255 sind möglich) und daneben die momentane prozentuale Größe des noch zur Verfügung stehenden Wortschatzes im Speicher.

Bei jeder Eintragung wird die Liste der Kreuzwörter gekürzt und das benutzte Wort aus der Liste gestrichen. Dies geschieht durch Vertauschen der Stringvektoren des zu streichenden Wortes und des letzten Wortes im Array (Zeile 300—399). Diese Methode wurde gewählt, um die Bildung von neuen Strings im Speicher zu vermeiden und der schrecklich langsamen Garbage-Collection aus dem Wege zu gehen.

Während der Rätselerzeugung können Sie sich in Ruhe überlegen, ob Sie ein automatisches Nachladen von Wortschätzen gestatten wollen oder nicht. In der rechten Bildhälfte ist inzwischen die Belegung der Funktionstasten zu sehen, und mit der F6-Taste schalten Sie die Autolader-Option ein oder aus. Erscheint »Aus« im F6-Tastenfeld, ist das Nachladen gesperrt.

Die Maschinenroutine sucht derweil ständig nach passenden Stellen im Wortfeld. Dieser Vorgang kann im Prinzip endlos sein, da irgendwann natürlich kein geeignetes Wort mehr zu finden ist. Aus diesem Grund hat das Programm eine »Geduld-Schwelle«, eine Anzahl von Suchversuchen, innerhalb derer ein passendes Wort gefunden werden muß. Ist die Versuchszahl ohne Erfolg abgelaufen, geht das Programm davon aus, daß der Wortschatz nicht mehr ausreicht. Diese »Geduld-Schwelle« können Sie in Zeile 964 selbst bestimmen.

Bevor allerdings in der obersten Bildzeile die Meldung »Wortschatz ungenügend« erscheint (Zeile 64/ Unterprogramm 110—158), hebt das Programm noch die »Blausperre« auf (Zeile 66). Dieses Flag hat dem Maschinenprogramm bisher mitgeteilt, daß nur dann eine Eintragung erlaubt ist, wenn dabei auch ein blaues, unbesetztes Feld abgedeckt wird. Eine Maßnahme, um das Rätsel möglichst dicht zu packen. Ab sofort ist also auch das Einpassen eines Wortes nur auf besetzten Feldern möglich. Läßt sich auch jetzt kein Wort mehr finden, erscheint endgültig der Hinweis auf mangelnde Wortauswahl. Wenn Sie das »Autoladen« zugelassen haben, wird nun der Folgewortschatz, falls vorhanden, gelesen und das Spiel beginnt von Neuem. Selbstverständlich ist auch das Laden von Wortdateien vor Ablauf der »Geduld-Schwelle« machbar. Dazu dient die F1-Taste. Aber Vorsicht, auf die Gefahr, daß Sie einen bereits verbrauchten Wortschatz nochmal laden, müssen Sie schon selbst achten.

Es dauert gar nicht so lange, dann ist das Wortfeld so dicht gepackt, daß der Maschinenroutine keine Eintragung mehr gelingt. Jetzt sind Sie gefordert, und mit der F7-Taste wählen Sie die Handeingabe an. Genau wie bei der Eintragung der ersten Startwörter können Sie Ihre »Lückenfüller« positionieren und die Fragestellung dazu eingeben.

Zufrieden mit Ihrer Arbeit (hoffentlich!) bleibt nur noch der Ausdruck des Rätsels. Mit F4 wird er gestartet. Das Unterprogramm für die Druckausgabe nimmt im Programm den weitaus größten Platz ein (Zeile 740—914). Ich verwende einen Epson RX-80-Drucker, der mit Hilfe einer speziellen Treibersoftware (Epson Software-Interface) auch CBM-Sonderzeichen drucken kann. Dazu muß eine unübliche Geräteadresse (6) angegeben werden (Zeile 761 und 900). Sollten Sie also einen CBM-grafikfähigen Drucker am IEC-Bus betreiben, tauschen Sie diese Adresse gegen die gewohnte »4« (im Listing schon geändert).

Auch die Druckerinitialisierung ist von Drucker zu Drucker verschieden (Zeilen 754—758). Achten Sie darauf, daß Ihr Drucker hier folgende Einstellung erhält:

— Zeilenabstand = 0

— CBM-Grafikmodus

— Startposition des Druckkopfes = lr (Linker Rand, kann in Zeile 946 geändert werden).

Für den Ausdruck der Fragestellung wird der RX80 im Engschriftmodus versetzt, um Platz zu sparen (Zeile 824—827). Auf diesen Effekt können Sie natürlich verzichten, müssen dann aber die Zeilenlänge (cz, in Zeile 948) ändern, da der Ausdruck vom Programm mit Randausgleich versehen wird (Zeile 847 — 889).

Nun wie versprochen zum Maschinenprogramm, kurz genannt »Such« (Bild 4 und 5). Diese Routine liegt im Bereich $C000/49152, wo sie gut gegen überschreiben durch Basic geschützt ist. Der Einsprung erfolgt über den USR-Vektor (definiert in Zeile 931 bis 932), das heißt es findet eine Parameterübergabe zwischen Basic und Maschinenroutine statt. Basic übergibt in »dl« die Anzahl der Versuche, die die Routine durchlaufen sollen (Zeile 44). Mit der Rückkehr aus der Routine wird der Variablen »u« ein Wert zwischen 1 und 4 zugewiesen. Aus dem Wert von »u« kann also auf die Ursache für den Abbruch der Routine geschlossen werden:

— u = 1; eine Taste ist betätigt worden.

— u = 2; die Suche nach einem passenden Kreuzwort war vergeblich.

— u = 3; ein Wort wurde gefunden und in das Wortfeld eingetragen.

— u = 4; die maximale Anzahl (255) von eingetragenen Kreuzwörtern ist erreicht; keine weitere Eintragung möglich. Das Rahmenprogramm kann jetzt entsprechend reagieren und zum Beispiel im Falle u = 3 das benutzte Wort aus der Wortliste streichen. Im Falle einer Eintragung (von Hand oder automatisch) wird das Wort nicht nur in den Bildspeicher eingesetzt, sondern es werden auch einige weitere Informationen abgelegt:

a) Eintragung im »Wortbeginn/ende«-Speicher (50176 bis 50575); hier wird vermerkt, ob ein Rätselfeld den Start- oder Endpunkt eines Kreuzwortes repräsentiert. Der Speicher ist, wie die folgenden auch, in 20 Zeilen mit je 20 Positionen (Speicher-

Fortsetzung auf Seite 161


64ER ONLINE



Fortsetzung von Seite 158

stellen) aufgeteilt. Das linke Halbbyte (4 Bit) einer Speicherstelle trägt die Informationen über senkrechte, das rechte Halbbyte über waagerechte Start/Endpositionen. Das 1. Bit im Halbbyte wird für Endpunkte gesetzt, das 3. Bit für Startpunkte. Diese Informationen werden später bei der Ausgabe des Rätsels auf dem Drucker benötigt, um an den richtigen Stellen die Nummer der zugehörigen Fragestellung eintragen zu können.

ne braucht dann nur den Paß des Wortes zu überprüfen und bei korrekter Eintragung ins Basic zurückzukehren. Paßt das Wort allerdings nicht, verfällt die Routine in den Automodus. Das Maschinenprogramm durchläuft im Automodus eine Schleife, die zunächst durch Auslesen des Rauschgenerators im Soundchip ein Wort aus dem Wortschatz per Zufall bestimmt. Dann erfolgt auf die gleiche Weise die Auswahl eines Startpunktes

Hinweise zum Eintippen

In diesem Listing wurden die meisten Steuerzeichen umgesetzt in Buchstabenkombinationen, die in eckigen Klammern stehen. Es bedeuten:

clr	=Shift Clear/Home-Taste	rvof	=Revers off
home	=Home-Taste	crsr	=Cursor rechts
cyan	=Control und 4	crsl	=Cursor links
whit	=Control und 2		(crsl*28 = 28 mal
red	=Control und 3		Cursor links)
rvon	=Revers on	crsd	=Cursor unten

In Zeile 51 bis 56 (Listing 3) bedeuten die Grafikzeichen (von oben nach unten) f7,f4,f1,f6,f8,f2.
Ein reverses »Z« bedeutet die Farbe Hellblau = Commodore-Taste und 7.

b) Eintragungen im »Senkrecht«-Speicher (50576 bis 50975); in dieser Speichermatrix werden die Ordnungsnummern der Fragestellungen für senkrechte Rätselwörter abgelegt. Bei der Druckausgabe wird diese Matrix abgefragt (Zeile 773 bis 777), um die Nummer der Fragestellung im entsprechenden Startfeld einzusetzen.

c) Eintragung im »Waagerecht«-Speicher (50976 bis 51375), wie unter b) Zeile 796 bis 800).

Um die Speicherinhalte vor dem Start des Rätselprogramms zu löschen, springt man die Routine mit »sys 50016« an (Zeile 937).

Nun zur Arbeitsweise der Routine während der automatischen Rätselerzeugung. Zunächst muß das Maschinenprogramm erst einmal wissen, ob es wegen einer Handeintragung angesprungen wurde oder zur automatischen Wortsuche. Dazu liest die Routine die Speicherstelle 26 aus. Ist das Ergebnis Null, wird in den Automodus verzweigt. Andernfalls wird in 26 die Länge des von Hand eingetragenen Wortes übergeben (siehe Zeile 662 bis 663). Die Routi-

im Bildspeicher. Ist dieser Punkt nicht geeignet für eine Eintragung, wird Zeile für Zeile des Wortfeldes nach einer Alternative gesucht. Im Falle eines Treffers startet der Wortvergleich. Das gewählte Wort wird mit dem Inhalt des Bildspeichers auf Übereinstimmung geprüft. Paßt es nicht, kommt das nächste Wort aus dem Wortschatz-Array an die Reihe. Bei Erfolg kehrt »Such« ins Basic zurück, wenn nicht, beginnt die Schleife von Neuem.

Damit der Zugriff auf das Array klappt, muß es nur als erstes im Basic-Programm definiert worden sein (Zeile 925). »Primitiv« werden Sie vielleicht anmerken. Richtig, aber Computer sind nun mal (sehr) schnelle Idioten.

(Gert Büttgenbach/gk)

```
bitparallel plottere
epromdndrurladerdoso
maabpfaradgfsrzeiger
yrkaobuchipcceespdpd
latnkasramqbasiocxiii
amodemderfwmlecadnxg
terzlprmdlesmthdatei
itvkreuzwortraetselt
oewaerceaptabankprha
nrgsgekirpycursorafl
 mosiccceydkbiaboka
gstesechecksumtjgtvs
amottnpeeklispzeriap
tawtetmnibbleuokavre
troerrekursivlmtmgii
ekrpioveditorsskmiac
returnstatuscobolrbh
lmatrixtastaturdelle
jhnsecfenstertreiber
hztmaschine  uniqegr
```

Lösung des Kreuzworträtsels

Programmierwettbewerb: Dokumentationshilfe

Insgesamt 1000 Mark zu gewinnen. Möchte man ein Programm analysieren oder schreiben, und die Dokumentation ist nicht oder nur mangelhaft vorhanden, ist eine automatische Dokumentationshilfe ein interessantes Werkzeug.

Die Aufgabe, die wir diesmal stellen, ist nicht nur eine Herausforderung an Programmierer, sondern soll zudem für Software-Entwickler ein nützliches Utility sein. Es geht um die Programmierung einer erweiterten Crossreferenzliste. Eine Crossreferenzliste durchsucht per Definition ein beliebiges Programm nach Variablen und Sprungbefehlen und gibt sie auf einem Drucker in gut lesbarer Form aus. Wir wollen aber in diesem Programmierwettbewerb ein vollständiges Werkzeug zur Dokumentation eines sich in der Entwicklung befindlichen oder fertigen Programms erhalten. Im einzelnen sollte das Programm folgendes können:

1. Alle Programmzeilennummern drucken, die Sprünge enthalten. Ausgegeben werden soll die Zeilennummer, dahinter die Zeilen, die angesprungen werden.

2. Ausgabe aller Programmzeilen, die angesprungen werden, wenn möglich mit den Zeilen, von denen aus der Sprung erfolgt.

3. Ausgabe aller im Programm verwendeten Variablen.

3.1 In der Reihenfolge, wie sie im Programm auftauchen.

3.2 In sortierter Reihenfolge: Sortiert nach Gruppe (Integer, Real, Strings und Felder) sowie alphabetisch.

3.3 In welcher Zeile sie definiert werden (Variable =) und in welcher Zeile sie benutzt werden (= Variable).

3.4 Es soll zu jeder Variable ein Kommentar eingegeben werden können.

4. Denkbar wäre auch, die ganze Prozedur innerhalb wählbarer Grenzen (zum Beispiel zwischen Zeile 1000 und 2000) eines Programms ablaufen zu lassen.

Wie Sie aus dem letzten Punkt ersehen können, sind den Ideen keine Grenzen gesetzt. Wichtig ist vor allen Dingen, daß ein komplettes Dokumentationsprogramm für die eigene Entwicklung und zur Analyse fremder Programme zustande kommt. So könnte eine automatische Aufschlüsselung nach Zeilennummern oder die Erstellung eines Fluß- oder Nassi-Shneidermann-Diagramms durchaus mit eingebaut werden. Lassen Sie Ihre Phantasie spielen und dokumentieren eigene und fremde Programme auf die bestmögliche Art und Weise.

Es wird mindestens zwei Gewinner geben: Einer für die beste Lösung in Basic, der andere für das beste Assembler-Programm.

Wenn Ihre Lösung von der oben genannten Aufgabenstellung etwas abweicht, so ist das keine Disqualifikation. Bewertungskriterien werden vor allem sein: Nutzbarkeit, Übersichtlichkeit, Schnelligkeit und Komfort.

Schicken Sie Ihre Lösung unter dem Stichwort »Programmierwettbewerb: Dokumentationshilfe« an folgende Adresse:

Markt & Technik Verlag AG, Redaktion 64'er, Hans-Pinsel-Str. 2, 8013 Haar bei München



Einmal im Monat gibt es die SUPERCHANCE

Diese nicht einmalige Gelegenheit sollten Sie nutzen. Wie? Schicken Sie uns Ihr bestes, selbst erstelltes Programm. Bei der Art des Programms sind wir nicht wählerisch.

Sie haben ein sehr gutes (Schieß-, Knobel-, Denk-, Action-, Abenteuer-) Spiel geschrieben: einschicken!

Sie verfügen über ein komfortables Disketten-Kopier-(Sortier) Programm mit einigen außergewöhnlichen Leistungsmerkmalen: einschicken!

Sie haben das Basic um einige sinnvolle Befehle erweitert: einschicken!

Sie arbeiten mit einem selbsterstellten Textverarbeitungsprogramm, einer eigenen Tabellenkalkulation, einem semiprofessionellen Datenverwaltungsprogramm: einschicken!

Sie zeichnen und konstruieren mit einem selbsterstellten Programm in hochauflösender Grafik: einschicken!

Wir freuen uns über jeden Beitrag und honorieren mit bis zu

2 000 Mark

für das Listing des Monats

Aus den besten Listings, die veröffentlicht werden, sucht die 64'er-Redaktion einmal im Monat das »Listing des Monats« aus. Alle Listings, die im 64'er abgedruckt sind, werden mit 100 bis 300 Mark honoriert. Die genaue Vorgehensweise beim Einsenden von Listings ist in »Wie schicke ich meine Programme ein?« Ausgabe 4/84 beschrieben:

Schicken Sie Ihr Listing an: Redaktion 64'er, Superchance: Listing des Monats, Hans-Pinsel-Str. 2, 8013 Haar bei München.

Wir suchen die Anwendung des Monats

Anwendung des Monats, was ist das? Nun, Sie haben einen Commodore 64 oder einen VC 20 und versuchen diesen irgendwie sinnvoll einzusetzen. Unter einer sinnvollen Anwendung versteht die 64'er Redaktion alles, was beispielsweise Programme im häuslichen Bereich bewirken. Es kann sich dabei um die Berechnung der Benzinkosten für Ihren Wagen handeln, um ein eigenes Textverarbeitungsprogramm gehen, sich um die Verwaltung Ihrer Tiefkühltruhe drehen oder ein ausgeklügeltes Telefon- und Adreßregister sein.

Setzen Sie Ihren VC 20/C 64 mehr oder weniger beruflich ein? Auch, oder vor allem, das ist eine sinnvolle Anwendung. Sie führen die Lohn- und Gehaltsabrechnung, Ihre Lagerverwaltung, die Bestellungen auf einem Commodore-Heimcomputer durch? So spezielle Anwendungen wie die Berechnung der Statik von selbstgezimmerten Regalen, von Klimadiagrammen oder Vokabellernprogrammen für den Schulunterricht oder die Zinsberechnung bei Krediten sind ebenfalls Themen, die mehr als konkurrenzfähig sind.

Uns ist die Anwendung des Monats

500 Mark

wert.

Schreiben Sie uns, was Sie mit Ihrem Computer machen:

Redaktion 64'er, Aktion: Anwendung des Monats, Hans-Pinsel-Str. 2, 8013 Haar bei München.

Fortsetzung von Seite 28
verlust — möglich; mit dem Cursor kann man auf das gewünschte Programm fahren und mit CTRL + L laden. CTRL + / liest den Fehlerkanal der Diskette. Das lästige »,8« nach dem LOAD- oder SAVE-Befehl darf entfallen, weil die Primäradressen 1 und 8, sich nun beide auf die Floppy beziehen.

Der Preis ist mit 295 Mark nicht gerade niedrig, aber dafür erhält man eine Lösung fast ohne Kompromisse. Mitgeliefert wird neben der Hardware eine Einbauanleitung und eine Diskette mit Kopierprogrammen. Als zusätzlichen Service bietet die Herstellerfirma an, daß im Falle von Überarbeitungen, jederzeit die alte gegen die neueste Version kostenlos umgetauscht werden kann.

Bedenkt man, daß die Entwicklung von »schnelleren« Soft- und Hardwarelösungen für die Floppy gerade erst begonnen hat, muß man sich über die Qualitäten dieser Produkte freuen. Interessanterweise kommen sie nicht über den großen Teich zu uns herüber. Auch wenn nur die Vorabversionen vorgeführt wurden, lassen sie doch in jedem Fall erkennen, daß wirkliche Könner dahinterstecken. Und sie wissen genau, daß, so sensationell ihre Entwicklung auch ist, die Konkurrenz nicht schläft. So wird ein Produkt mit 98%iger Kompatibilität bei einer sechsfach höheren Geschwindigkeit erfolgreicher sein, als 80%ige Kompatibilität bei einer fünfzehnfach schnelleren Ladezeit. Sicher wird in dieser Hinsicht noch einiges auf uns zukommen. Man kann gespannt sein.

(M. Kohlen/ D. Weineck/gk)

Bezugs- und Informationsquellen: Roßmöller GmbH, Finkenweg 1, 5309 Meckenheim

S & S-Soft, Schöttelkamp 23a, 4620 Castrop-Rauxel 9

Optronik Service International, Salzdahlumer Str. 196, Postfach, 3300 Braunschweig

Inserentenverzeichnis



Impressum

Herausgeber: Carl-Franz von Quadt, Otmar Weber

Chefredakteur: Michael M. Pauly (py)
Stellv. Chefredakteur: Michael Scharfenberger (sc)
Redakteure: aa = Albert Absmeier, leitender Redakteur, ev = Volker Everts, gk = Georg Klinge, rg = Christian Rogge
Redaktionsassistenz: Gerda Siegl (202)
Fotografie: Janos Feitser, Titelfoto: Alex Kempkens

Layout: Leo Eder (Ltg.), Dagmar Berninger, Willi Gründl, Cornelia Weber

Auslandsrepräsentation:
Schweiz: Markt&Technik Vertriebs AG, Alpenstrasse 14, CH-6300 Zug, Tel. 042-223155/56, Telex: 862329 mut ch
USA: M&T Publishing, 2464 Embarcadero Way, Palo Alto, CA 94303; Tel. 001-4240600; Telex 752351

Manuskripteinsendungen: Manuskripte und Programmlistings werden gerne von der Redaktion angenommen. Sie müssen frei sein von Rechten Dritter. Sollten sie auch an anderer Stelle zur Veröffentlichung oder gewerblichen Nutzung angeboten werden, so muß dies angegeben werden. Mit der Einsendung von Manuskripten und Listings gibt der Verfasser die Zustimmung zum Abdruck in von der Markt & Technik Verlags AG herausgegebenen Publikationen und zur Vervielfältigung der Programmlistings auf Datenträger. Honorare nach Vereinbarung. Für unverlangt eingesandte Manuskripte und Listings wird keine Haftung übernommen.

Herstellung: Klaus Buck (180)

Anzeigenverkauf: Brigitta Fiebig (211)

Anzeigenverwaltung und Disposition: Michaela Hörl (171)

Anzeigenformate: 1/1-Seite ist 266 Millimeter hoch und 185 Millimeter breit (3 Spalten à 58 mm oder 4 Spalten à 43 Millimeter). Vollformat 297x210 Millimeter. Beilagen und Beihefter siehe Anzeigenpreisliste.

Anzeigenpreise: Es gilt die Anzeigenpreisliste Nr. 1 vom 1. März 1984.
Anzeigengrundpreise: 1/1 Seite sw: DM 7400,-. Farbzuschlag: erste und zweite Zusatzfarbe aus Europaskala je DM 1000,-. Vierfarbzuschlag DM 3000,-. Plazierung innerhalb der redaktionellen Beiträge: Mindestgröße 1/2-Seite
Anzeigen im Einkaufs-Magazin: Die ermäßigten Preise im Einkaufs-Magazin gelten nur innerhalb des geschlossenen Anzeigenteils, der ohne redaktionelle Beiträge ist. 1/1-Seite sw: DM 5400,-. Farbzuschlag: erste und zweite Zusatzfarbe aus Europaskala je DM 1000,-. Vierfarbzuschlag DM 3000,-. **Anzeigen in der Fundgrube: Private Kleinanzeigen** mit maximal 5 Zeilen Text DM 5,- je Anzeige. **Gewerbliche Kleinanzeigen:** DM 10,- je Zeile Text.
Auf alle Anzeigenpreise wird die gesetzliche MwSt. jeweils zugerechnet.

Vertriebsleitung, Werbung: Hans Hörl (114)

Vertrieb Handelsauflage: Inland (Groß-, Einzel- und Bahnhofsbuchhandel) sowie Österreich und Schweiz: Pegasus Buch- und Zeitschriften-Vertriebs GmbH, Plieninger Straße 100, 7000 Stuttgart 80 (Möhringen), Telefon (0711) 72004-0

Erscheinungsweise: 64'er, Magazin für Computerfans, erscheint monatlich, Mitte des Vormonats.

Bezugsmöglichkeiten: Leser-Service: Telefon 089/4613-119. Bestellungen nimmt der Verlag oder jede Buchhandlung entgegen. Das Abonnement verlängert sich zu den dann jeweils gültigen Bedingungen um ein Jahr, wenn es nicht zwei Monate vor Ablauf schriftlich gekündigt wird.

Bezugspreise: Das Einzelheft kostet DM 6,-. Der Abonnementspreis beträgt im Inland DM 72,- pro Jahr für 12 Ausgaben. Darin enthalten sind die gesetzliche Mehrwertsteuer und die Zustellgebühren. Der Abonnementspreis erhöht sich um DM 18,- für die Zustellung im Ausland, für die Luftpostzustellung in Ländergruppe 1 (z.B. USA) um DM 38,-, in Ländergruppe 2 (z.B. Hongkong) um DM 58,-, in Ländergruppe 3 (z.B. Australien) um DM 68,-.

Druck: Druckerei E. Schwend GmbH, Schmollerstr. 31, 7170 Schwäbisch Hall

Verantwortlich: Für redaktionellen Teil: Michael M. Pauly.
Für Anzeigen: Hannelore Schmidt.

Vorstand: Carl-Franz von Quadt, Otmar Weber

Anschrift für Verlag, Redaktion, Vertrieb, Anzeigenverwaltung und alle Verantwortlichen:
Markt&Technik Verlag Aktiengesellschaft, Hans-Pinsel-Straße 2, 8013 Haar bei München, Telefon 089/4613-0, Telex 522052

Telefon-Durchwahl im Verlag:

Wählen Sie direkt: Per Durchwahl erreichen Sie alle Abteilungen direkt. Sie wählen 089-4613 und dann die Nummer, die in Klammern hinter dem jeweiligen Namen angegeben ist.



VORSCHAU

Softlearning

Die Lehr- und Lernmethoden an den Schulen ändern sich laufend. Auch auf dem Gebiet des computerunterstützten Lernens können neue Wege beschritten werden. SM hat sich bei der Auslegung ihrer Lernsoftware an dem Erfolgskonzept des Superlearning, das bei der Schulung von Führungskräften in der Wirtschaft seit zwei Jahren für Furore sorgt, orientiert. Also Lernen mit dem C 64, ohne blockierende Ängste und Streßgefühle.

Alles über Strings

Fast kein Programm kommt ohne Strings aus. Was ist eigentlich ein String? Warum gibt es manchmal »Zwangspausen« — die Garbage Collection — beim Verarbeiten von vielen Strings und wie kann man sie umgehen? Die Antworten und viele Tips und Tricks lesen Sie in der nächsten Ausgabe.

Forth to Clarity

Voran zum Verständnis — nämlich zum Verständnis von Forth, dieser eigenwilligen Programmiersprache, die in letzter Zeit immer größere Verbreitung findet. Anhand eines konkreten Programmierbeispiels, nämlich dem Bau eines Decompilers, wird in leicht nachzuvollziehender Form in die grundsätzliche Struktur der Sprache eingeführt.

Hypra-Load mal fünf

Das in Ausgabe 10 vorgestellte Hypra-Load war eine Sensation. Aus diesem Grund bringen wir viele Anregungen und Verbesserungen zu diesem Programm. Außerdem erfahren Sie, wie Hypra-Load fest in das Betriebssystem eingebaut werden kann.

Außerdem...

- Hi-Eddi, ein faszinierendes Zeichen- und Grafikprogramm als Listing des Monats
- ein strategischer Handballtrainer als Anwendung des Monats
- sieben Kurse
- und wieder viele Tips und Tricks für C 64 und VC 20.

Der Nachfolger

Der C 16, ursprünglich als Nachfolger des VC 20 konzipiert, ist dabei, sich einen eigenen Platz an der Sonne im heißumkämpften Markt der kleinen Computer zu sichern. Sein Basic 3.5 mit mehr als 75 Befehlen unterstützt Grafik, Sound und strukturierte Programmierung. Ist der C 16 wirklich nur als Nachfolger des VC 20 gedacht, oder kann er sogar dem C 64 das Leben schwer machen? Unser großer Testbericht zeigt, was der C 16 wirklich kann — und wer ihn sich kaufen sollte.



Assembler im Test

Immer mehr C 64-Besitzer wollen mehr als nur in Basic programmieren. Um jedoch effektiv in Maschinensprache arbeiten zu können, braucht man einen Assembler. Wir haben die bekanntesten und leistungsfähigsten für Sie herausgesucht und getestet, darunter MAE, Profimat, Pofisoft, ASS64, ASSI und TE.X.AS. Lesen Sie, welcher für Sie am geeignetsten ist.

G-Basic, die Spracherweiterung für alle Fälle

Spracherweiterungen für den C 64 gibt es schon eine ganze Menge. Die bisher bekannteste dürfte wohl Simons Basic sein. Hat ein neues Produkt wie G-Basic eine Chance? Was leistet G-Basic? Gibt es Probleme oder Fehler? Ist es mit 259 Mark preiswert oder wieder nur eine Lösung mit Kompromissen?

Digitalisierer

Digitalisierer verwandeln Videobilder oder Sprachimpulse in für den Computer verständliche Signale. Der Empfang von Satellitenbildern, die Auswertung durch den Computer und die Ausgabe auf einem Farbdrucker ist nur ein Beispiel für die Anwendung. Lassen Sie sich überraschen, was man mit Digitalisierern noch alles machen kann.

